09/720005

PCT/AU99/00501

AUSTRALIA

AU99/501

PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Patent Office
Canberra 5

I, KIM MARSHALL, MANAGER PATENT OPERATIONS, hereby certify that the annexed is a true copy of the Provisional specification in connection with Application No. PP 4164 for a patent by PAUL GUIGNARD and SAVANT SYSTEMS INTERNATIONAL PTY LTD filed on 18 June 1998.

COMMONWEALTH OF AUSTRALIA
AUSTRALIA
PATENT OFFICE

WITNESS my hand this Nineteenth day of July 1999

KIM MARSHALL
MANAGER PATENT OPERATIONS

PROVISIONAL SPECIFICATION

GENERIC KNOWLEDGE MANAGEMENT SYSTEM

TABLE OF CONTENTS

Part 1: Introduction

PROVISIONAL SPECIFICATION

GENERIC KNOWLEDGE MANAGEMENT SYSTEM

TABLE OF CONTENTS

Part 2: Generic Knowledge Management System (GKMS)

PROVISIONAL SPECIFICATION

GENERIC KNOWLEDGE MANAGEMENT SYSTEM

TABLE OF CONTENTS

Part 3: GKMS Implementation

PROVISIONAL SPECIFICATION

GENERIC KNOWLEDGE MANAGEMENT SYSTEM

TABLE OF CONTENTS

Part 4: GKMS Applications and Examples

Part 1: Introduction

1. SUMMARY

Knowledge-based systems (KBSs) are a class of programs in which the knowledge about a range of problems and their solutions is stored separately from the code that is used to manipulate that knowledge to produce solutions. KBS design is different from traditional software where the knowledge about a problem and its solution is integrated into the code that processes that knowledge. Because of the separation between knowledge and its processing KBSs, in principle, should be easier to develop and maintain and safer to use than traditional programs. However it is not the case in practice and the penetration of KBSs in industry is very small compared to spreadsheets, databases and word processors. This is despite the enormous appeal and promise of using knowledge to make organisations more resilient, flexible and to increase productivity.

This specification describes an alternative technology for the design of knowledge systems, for the acquisition, modelling and storage of knowledge for these systems, and for the maintenance of that knowledge. This alternative technology, referred to below as GKMS (Generic Knowledge Management System) offers a solution to the knowledge acquisition and the knowledge maintenance problems. In addition, knowledge systems designed with this new technology: a) know the limit of their knowledge, b) restrict the use of their knowledge to situations they recognise and that have been approved by human experts, c) only provide advice based on what they know, d) inform the users when the limit of the knowledge available is detected, and e) when this happens, they ask domain expert(s) to extend the knowledge available in the system.

Any software system (as any artifact) embodies knowledge. In traditional software this knowledge is typically hardcoded in the system. With the technology described in this specification, software for any application can be expresssed and developed as a GKMS. That is the knowledge for the software can be expressed separately from the code that executes it, with the advantages described in the previous paragraph.

With GKMS, non computer specialists (managers and domain experts) can store, access, control and use knowledge in the same way that they now store, access, control and use information. They can also develop software which, up to now, could only be developed by software engineers. GKMS technology enables organisations and people to move from information-based operations to knowledge-based operations. The expected practical and commercial impact on personal and business processes and effectiveness is very significant.

2. OVERVIEW OF KNOWLEDGE MANAGEMENT TECHNOLOGY

2.1 COMPARISON OF EXISTING KNOWLEDGE TECHNOLOGIES

In this section we review four well known technologies that are used for knowledge management: expert systems, case-based reasoning (CBR) systems, decision trees and lookup tables. We also consider standard software development and the technoloogy disclosed in this specification. The comparison of these technologies, shown in Table 1, is a qualitative assessment based on ten attributes:

Knowledge model

The model used for representing and processing knowledge. Answers the question: how is the knowledge embodied in the system expressed? The model determines the key performance features of a system.

Knowledge acquisition difficulty

Experience shows that acquiring knowledge from experts is difficult. An effective and convenient way of acquiring knowledge would open new applications, make knowledge based technology much more appealing to the market and increase its penetration.

Knowledge maintenance difficulty

Relates to the cost of operating a system over its life during which new knowledge is required to reflect changing conditions or usage. Knowledge maintenance difficulty is related to knowledge acquisition difficulty.

Context for the knowledge

Is the context for the knowledge explicit? Context needs to be explicit for adequate knowledge use and control. For example, in a medical situation, the type of questions to ask depend on whether one takes a specialist view of health (ie: cardiology) or a more holistic view (re: cardiology and life style). The issues one needs to enquire about in the latter case are different from the former case and the outcomes (treatments, recommendations) are also likely to be different.

Knowledge processing

The type of processing that takes place. Is it deterministic? Can it introduce uncertainties or errors (not due to software errors but to the processing technique adopted)?

Knowledge certification

Can the knowledge be certified? That is, can an expert be held accountable for the quality of the knowledge? Knowledge certification is essential for controlling the quality of the knowledge and make its use safe.

Does the system know the limits of its knowledge?

A system that does not know the limit of its knowledge is not safe to use by people who are non-experts and who may not recognise when the system provides inaccurate or inappropriate advice.

Level of domain expertise required of users

Do users need to have knowledge about the field of application to be able to use the system safely or is the system safe, even for non-experts.

Level of computer expertise required of domain experts

Do domain experts need to be computer knowledgeable or can they simply be experts in their chosen application domain (that may have not connection with computing).

Can system be used with limited knowledge in its knowledge base?

This is a significant advantage as it enables systems to be fielded early with additional knowledge added later, in response to user requirements.

Table 1: Comparison between expert systems, CBR, decision trees, lookup tables, "standard" software and GKMS technology

	Feature	Importance	Expert systems	Case based reasoning (CBR)	Decision trees	Lookup tables	Standard software	GKMS technology
1	**Knowledge model**	A model close to the way experts and users think enables them to feel comfortable with the system's operations	• Self contained re-usable, context independent logical clauses (rules) • Context expressed as (other) rules • Difficult to relate to (semantic gap)	• Situations defined as cases • Attaches outcomes to cases • Easy to relate to	• Decision tree • Describes the situation with more and more precision • Some knowledge may need to be duplicated in several branches	• Knowledge described as attributes in a table	• Knowledge embedded in the code • Knowledge and data are not separated	• Knowledge described as regions (or patterns) in a source (or problem) space linked to outcomes in the destination (or solution) space
2	**Knowledge acquisition difficulty**	Determines the cost of development and the range of cost effective applications	• Difficult • Time consuming • Costly	• Convenient	• Simple for small knowledge bases • Becomes quickly complicated	• Convenient • Combinatorial explosion as number of attributes grows	• Related to the software development process	• Convenient • Can be prompted by enquiries • Simple to relate to (no semantic gap)
3	**Knowledge maintenance difficulty**	Determines the cost over the life of the system	• Difficult • Time consuming • Costly	• Convenient	• Difficult as a change at the root a branch can affect the whole branch	• Convenient	• Convenient	• Convenient • Supported in the same way as knowledge acquisition

	Feature	Importance	Expert systems	Case based reasoning (CBR)	Decision trees	Lookup tables	Standard software	GKMS technology
4	**Knowledge context**	Determines the quality of the knowledge and whether its use can effectively be controlled	• Not explicit • Difficult to modify	• Not explicit • Can evolve	• Not explicit • Difficult to modify	• Not explicit • Difficult to modify (needs changing the table)	• Not explicit • Difficult to modify (needs changing code)	• Explicit • Can evolve and be controlled by experts and users
5	**Knowledge processing**	Is it deterministic, can it introduce effects (errors) unintended by expert?	• Based on recursive inferencing of logical clauses • Can follow many unpredictable paths	• Based on similarity measures with previous cases • Difficult to predict and control	• Ask questions that guide the user down a path in the tree	• Match of query with attributes in each row of the table (row must contain query)	• Run the software	• Knowledge elements compatible with enquiry fire • Support inferencing
6	**Knowledge certification**	Certification promotes accountability among experts and confidence of use among non-experts	• Very difficult as system outcomes depend on experts identifying correct knowledge blocks (rules), on these rules being correctly implemented in the system and on the system not producing unforeseen outcomes	• Certification not possible because the concept of similarity used in knowledge processing	• Difficult • Not implied in model	• Possible • Expert could certify each row and its outcome • However knowledge processing defeats this approach	• Related to software validation and certification (difficult and time consuming)	• Knowledge is certified by experts • GKMS only uses knowledge items in situations for which they have been approved by experts

	Feature	Importance	Expert systems	Case based reasoning (CBR)	Decision trees	Lookup tables	Standard software	GKMS technology
7	**Does system know limits of its knowledge**	Important for non-experts who may not recognise whether an output is correct or not	• Not implied in knowledge model • Requires extensive, difficult and time consuming testing • Difficult and unreliable	• Not implied in knowledge model • Relies on similarity rating (based on scores) • Unreliable	• No • Not implied in model	• Yes • Given by 'cases' described in the rows	• No, not explicitly	• Yes • Derives from the knowledge model in GKMS
8	**Domain expertise required of users**		• Medium • Needed to detect inappropriate outcomes	• Medium • Needed to evaluate similarity of cases	• Medium • Needed to check validity of outcomes	• Medium • Expertise needed to select correct outcome out of those offered by system	• Low or medium	• Low • Interactively simple • Users receive certified advice
9	**Computer expertise required of experts**	Can experts focus on their domain or do they need to know about computing	• Low, but knowledge engineer required for knowledge acquisition and maintenance	• Low	• Low	• Low	• High • Need to be able to develop software	• Low • Interactively simple • Easy to express one's expertise in GKMS
10	**Can system be used with limited knowledge in its knowledge base?**		• No • System must be packaged in its near final form before release	• Yes, but not safe for non-experts • Safe for experts who need to recognise whether an outcome is applicable	• No • Outcome not reliable	• No • Outcome not reliable	• No	• Yes • Safe (see point 7) • New knowledge can be added without affecting current knowledge

2.2 OVERVIEW OF KNOWLEDGE TECHNOLOGY

A large and well known class of knowledge systems are Expert Systems (ESs) which, in some limited domain and understanding of the term, can operate and perform in a way that is associated with human experts exercising their specialist knowledge. ESs can diagnose problems and make recommendations, by encoding the knowledge and reasoning skills of human experts related to a (usually quite) restricted domain of applications. The objective of expert systems technology is to enable managers or domain experts who are not computer specialists to enter their knowledge into a system that will then operate interactively with humans and provide recommendations about situations and problems that fall within its domain of expertise. Expert systems are usually rule-based systems in which knowledge is encoded in the form of (IF ... THEN...) rules. Unless otherwise stated, the term expert systems will refer below to expert systems of the rule-based type.

Current interest in knowledge-based systems comes from the realisation that this technology makes possible, in principle, the development of programs that support the capture, storage, use and maintenance of knowledge. These programs could give users access to, and control of, knowledge, in the same way that database management systems (DBMS) give access to, and control of, information. Users who are not domain experts could gain access to, and use, expert knowledge. Knowledge-based systems, correctly applied, have the potential to help organisations improve their productivity and profitability substantially.

In the past few years, expert systems technology has been successfully applied to a large range of problems in nearly all fields (academia, government, defence, all industry sectors). A significant number of developers and vendors of expert systems technology and tools now exist (Harmon et al, 1988; Boose 1989; AI Expert System Resource Guide, 1992; Mettrey, 1991) and they supply a large worldwide market.

Despite this success, expert system technology is being hampered in its large scale development and application by the difficulty of acquiring and maintaining the expert knowledge that enables expert systems to function (Hickman, 1986; Sestito & Dillon, 1994; Chien & Ho; 1994; Nwana, 1994; Olson et al, 1994; Mechitov et al, 1994). This is despite significant effort in the development of tools to assist with these tasks (Boose, 1989; Kingston, 1992). Specifically, knowledge needs to be acquired from human domain experts[1], modelled and coded into a program before an expert system can operate. Once this is done, knowledge in the expert system must be maintained so as to reflect the evolution of knowledge and the evolving requirements of real world applications. The two processes of knowledge acquisition and maintenance on which current expert system technology relies are difficult and time consuming, therefore costly.

These two problems largely explain why expert system technology has not gained widespread use the way databases, spreadsheets and word processors have. This is despite the enormous appeal and promise of being able to use knowledge to increase productivity.

[1] A domain expert is a recognised expert in a specific domain and whose decisions in this domain are accepted as being correct or as representing the state-of-the-art in the domain.

Expert systems remain a specialist technology that is used for specific applications, after a careful evaluation has shown the technical feasibility of the application and demonstrated an adequate return on investment.

Current expert systems technology has other limitations that further contribute to its limited use. Expert systems are difficult to verify and validate, they do not know the limit of their knowledge and their performance deteriorates dramatically when applied to situations for which their knowledge is not complete (experts systems performance deteriorate much more rapidly than human experts in similar situations). Partial solutions to these problems have been proposed for knowledge acquisition, such as machine learning using induction (Sestito & Dillon, 1994), connectionist systems (Mozer & Rambot, 1987; Honavar & Uhr, 1990) and "Ripple Down Rules" (Compton et al, 1993). The problem of verification and validation has also been extensively researched (Lee & O'Keefe, 1994; Int J of Intelligent Systems, Vol 9, No 8, special issue; Jafar & Bahill, 1988). While the techniques proposed are effective in some situations, they are not sufficient to overcome the practical limitations of the technology stated above.

This specification describes an alternative technology for the design of knowledge-based systems, for the acquisition, modelling and storage of knowledge for these systems, and for the maintenance of that knowledge. This alternative technology, referred to below as Savant technology, is based on a cognitive model of the way humans carry out much of their problem solving tasks. Savant technology offers a solution to the knowledge acquisition and the knowledge maintenance problems. In addition, knowledge systems designed with this new technology: a) know the limit of their knowledge, b) restrict the use of their knowledge to situations they recognise and that have been approved by human experts, c) only provide advice based on what they know, d) inform the users when the limit of the knowledge available is detected and e), when this happens, they ask domain expert(s) to extend the knowledge available in the system. These features are very significant and offer important theoretical and practical advantages over current expert system technology.

With this new technology, non computer specialists (managers and domain experts) can store, access, control and use knowledge in the same way that they now store, access, control and use information. This technology enables organisations and people to move from information-based operations to knowledge-based operations. The expected practical and commercial impact on personal and business processes and effectiveness is immense.

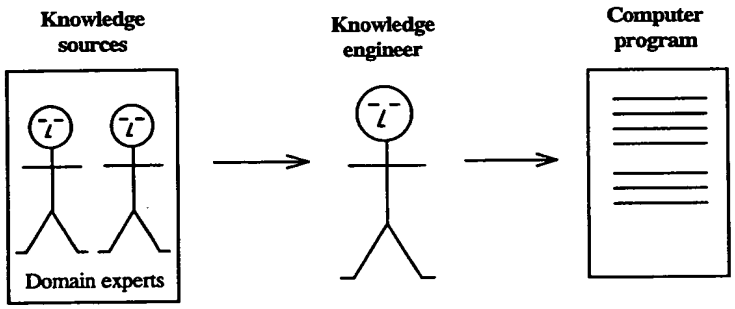
This specification comprises five sections: a) an overview of current knowledge-based system technology relevant to the presentation and evaluation of the new technology disclosed in this specification, b) a presentation of the cognitive model underlying Savant technology, c) the features and benefits associated with the cognitive model, d) the design methodology for an Savant system and e) architecture and implementation considerations.

2.3 OVERVIEW OF RELEVANT EXISTING KNOWLEDGE TECHNOLOGY

2.3.1 Knowledge acquisition

Knowledge Acquisition is the transfer and transformation of problem-solving expertise from some knowledge source to a program (Buchanan et al. 1983). Knowledge acquisition covers several stages: identification, conceptualisation, formalisation, implementation and testing (Buchanan et al. 1983). The knowledge acquisition process is difficult and typically beyond the technical capabilities of domain experts. It has been necessary to train a new type of specialist, the "knowledge engineer", to assist with this task. When knowledge comes from a human source, the knowledge acquisition process can be represented as illustrated below (Sestito & Dillon, 1994):

Figure 1: The knowledge acquisition process from domain experts



Knowledge engineers acquire knowledge from domain experts and other sources such as books, manuals, reports, case studies, etc. Below we concentrate on knowledge acquisition from human sources because most expert systems rely on domain experts for part and often all of their knowledge. A second reason is that only when human experts will be able to conveniently and effectively store and manipulate their knowledge in computer systems (that is when knowledge acquisition from human experts becomes convenient and effective) will expert systems become widely adopted by humans. This is because expert systems will only then become natural repositories and extensions of human cognitive abilities.

A source of the knowledge acquisition problem is the difficulty domain experts have at expressing the knowledge they possess, the way it is structured and the way they use this knowledge (Honavar & Uhr, 1990; Johnson-Laird, 1989, Nwana et al, 1994). This is referred to as the "knowledge mismatch" problem (Sestito & Dillon, 1994). The difficulty is compounded by problems of knowledge modelling, representation, implementation and testing. Numerous acquisition techniques and tools have been developed to try to alleviate the problem (Boose, 1989), without producing the breakthrough required for widespread commercial application of the technology.

2.3.2 Knowledge modelling and representation

Knowledge representation in expert systems is a symbolic representation based on production rules. The primitives for knowledge representation and processing are represented by symbols. A production rule is a "condition-action" pair expressed in the form of (IF ... THEN...). The first "..." represents a set of conditions between symbols and their possible values, and the second "..." is a statement about the consequence of the set of conditions being true. A rule fires only when its conditions are met.

Knowledge, in expert systems, is stored in a knowledge base. It is separate from the processing of knowledge which is controlled by an inference engine.

2.3.3 Inferencing (forward and backward chaining)

Reasoning in rule-based systems relies on the first-order predicate calculus (Levesque, 1986). There are three reasoning paradigms: forward, backward and opportunistic reasoning (Buchanan & Smith, 1988).

'A forward-chaining systems starts with a collection of facts and draws allowable conclusions, adding those to the collection and cycling through the rules... Backward reasoning is goal-directed and does not require all relevant data to be available at the time inferences begin... A backward-chaining system starts with a hypothesis (goal) to establish and asks, in effect, "what facts (premise clauses or rules) would need to be true in order to know that the hypothesis is true?"... Opportunistic reasoning combines some elements of both data-directed (forward) and goal-directed (backward) reasoning.' (Buchanan & Smith, 1988).

A well known problem of rule-based systems, related to inadequate reasoning and knowledge representation, is the brittleness of these systems. Unlike human experts, the performance of expert systems falls drastically as they encounter problems beyond the scope of their expertise. They lack robustness (Buchanan et al, 1990; Devedzic & Velasevic, 1990). This shortcoming coupled with the difficulty of acquiring knowledge and of verifying and validating expert systems make their use in many sensitive or critical applications difficult to justify or costly to implement.

In summary, expert systems of the rule-based type are characterised in the following way:

- Their knowledge is difficult to acquire, model and verify.
- Their knowledge is difficult to maintain.
- They do not know the extent of their knowledge.
- They have limited reasoning ability.
- They cannot restrict the use of their knowledge to approved situations, that is it is difficult to certify this knowledge (i.e.: they can apply their knowledge to situations for which the knowledge was not designed for)

- When they apply their knowledge to inappropriate situations, their performance degrades sharply and their conclusions can be totally inappropriate, sometimes to a dangerous extent.

Overall, and despite their success in specific areas, rule-based systems rely on a technology that makes their widespread, cost-effective acceptance and use difficult.

2.3.4 Case-Based Reasoning

Case-based Reasoning (CBR) is an attempt at approaching knowledge acquisition and processing from a different angle. CBR relies on the episodic model of memory in humans proposed by Tulving (1972, 1983) which was then applied to artificial intelligence by Schank (1972, 1975), Schank & Kolodner (1979), Kolodner (1980), Kolodner & Cullingford (1986), Kolodner (1993), Eloranta (1991). In CBR, knowledge is not modelled as a set of rules but represented as a set of episodes. These episodes correspond, according to CBR proponents, to the way humans use past experiences to support their reasoning. When a new problem is encountered, old episodes that are similar to the new problem are retrieved. These episodes and their answers are then modified, if necessary, to match the problem at hand. More specifically, the CBR process model comprises 6 steps (Riesbeck & Bain, 1987; Slade, 1991; Ketler, 1993):

1.	Assign indices	Identify key features characterising the new event and assign indexes to these features.
2.	Retrieve	The indexes are used to retrieve from memory a similar past case and its solution.
3.	Modify	The retrieved case and solution are modified to conform to the new situation, resulting in a new proposed solution.
4.	Test	The proposed solution is tried out.
5.	Assign and store	If the solution succeeds, indexes are assigned and a working solution stored.
6.	Explain, repair & test	If the solution fails, then explain the failure, repair the working solution and test again.

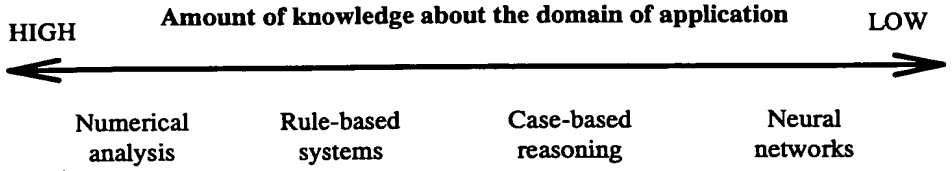
The indexing rules in CBR identify the predictive features in the input that provide appropriate indexes into the case (episodic) memory, which is the database of experience. Cases are retrieved on the basis of a similarity metrics that determines which cases most closely match the new case. The selection of adequate features for indexing is a persistent problem in CBR (Seifert et al, 1994).

CBR is most useful in establishing a partnership between the computer and a user. CBR usually addresses problems that are not well structured or that are difficult to define formally, that are open-ended and which have ill-defined concepts (Kolodner, 1993). These problems are best described in natural (or pseudo-natural) language. Typically, with CBR the user must have the expertise to a) determine whether the most similar case

retrieved matches the new problem, b) if not, how to modify the old case, c) how to test the new case, and d) how to describe it with indexes for storage in memory.

CBR fits in the spectrum of knowledge-based technologies as proposed by Mott (1993). The Figure below adapted from his work.

Figure 2: Technology continuum



Thus CBR is complementary to rule-based and both have their uses depending on the characteristics of the applications. Neural nets, mentioned in Section 1, are most useful when knowledge about the domain is low or difficult to formalise. Neural net technologies are not discussed further here as Savant does not fit at the "neural networks" end of the spectrum in the above Figure.

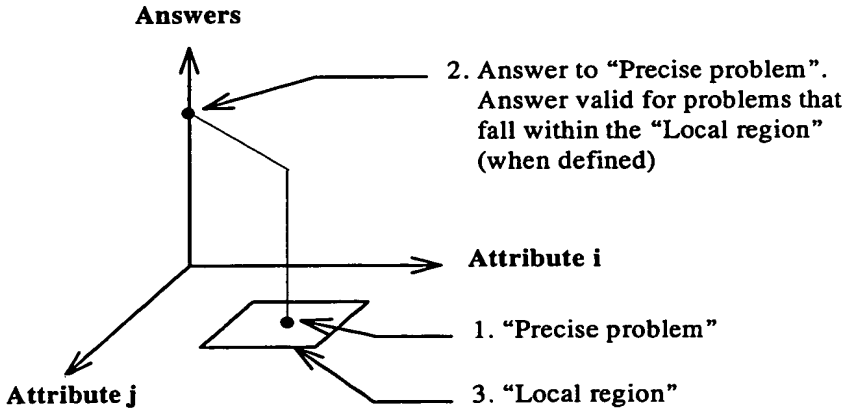
Part 2: Generic Knowledge Management System (GKMS)

1. COGNITIVE MODEL UNDERLYING GKMS TECHNOLOGY

1.1 THE "SOLVE SPECIFIC PROBLEM AND GENERALISE" COGNITIVE MODEL

GKMS is based on a cognitive model of a significant part of human beings' problem solving activities. It is hypothesised that humans frequently apply their expertise locally and then attempt to generalise. That is, humans look for and produce answers to problems that are clearly specified. Once this is done for a problem, they then attempt to generalise the validity of the result obtained to situations that are similar to the problem already encountered. With experience and when appropriate, the domain of applicability of the answer is broadened, from the single initial precise instance of the problem to a local region. That is, the initial answer is then deemed to apply to all situations that fall within the local region. When a new case is encountered that does not fall within a region, then a new answer is produced and, when appropriate, a new local region is defined. Regions can have varying sizes and shapes in the application space, they can overlap or be totally disjointed. This process is illustrated in Figure 3 below with a single specific problem and a single region. This cognitive model is described as the "Solve Specific Problem and Generalise" (SSPG) model.

Figure 3: Illustration of the SSPG model for humans' problem solving strategy



In Figure 3, the horizontal plane represents the problem space, that is, the space of all the attributes that are necessary to specify the problem. In most situations, human experts determine which attributes are relevant to what classes of problems. The vertical axis in Figure 3 represents the answer space, that is the space of all possible answers. The answer space, like the problem space, is a multi-dimensional space. The human problem solving strategy described in this Section encompasses both the problem and the solution space, which jointly form the application space. Problem solving, according to the model, proceeds as follows:

1. Produces or define a specific problem.

2. Produce an answer to the specific problem.

3. Generalise, that is, expand, when appropriate, the domain of validity of the answer given under 2 above to a "local region".

Once a region and its outcome (answer) have been produced by the domain expert, this knowledge can be used by users who are not domain experts. If a problem falls within the region, its answer is the outcome attached to the region.

Knowledge, according to this model, is inseparable from its domain of applicability. The problem space in Figure 3 represents the context for the description of the problems, and the solution space represents the context for the description of the solutions. These two contexts are explicit in the model. As knowledge is highly context dependent, this feature has vital practical applications. For example, general practitioners have different contexts (problem and solution contexts) for their knowledge from, say, psychiatrists. One cannot assess the validity and usefulness of their knowledge unless one knows about their respective contexts.

The cognitive model (problem solving strategy) outlined above is labelled the **"Solve Specific Problem and Generalise" (SSPG) model**. It is presented as a hypothesis although is appears can draw from the same sort of cognitive support as Case Based Reasoning (this is not pursued further here). This model is proposed as one of several needed to describe the whole range of human cognitive functions. It is clear that the SSPG model presents an important opportunity for technology development.

There are two ways for human beings to operate according to the SSPG model presented above.

a) When faced with a new problem, humans look for an old problem and then "extend" the old problem to a "local region". They then decide whether the new problem fits inside the region. If it does they use the answer corresponding to the old problem. If it does not, they generate a new answer.

b) Humans define a "local region" every time a new problem is encountered and an answer produced. That is, they attempt to define a region every time they generate a new answer. According to this, human beings, consciously or subconsciously, seek to understand the applicability of a new solution (that is, its sensitivity to small changes in the problem definition).

It seems reasonable to assume that understanding a problem domain means understanding answers to different classes of problems in the domain. The second way is then favoured when experts have a broad knowledge of the problem domain.

c) In practice, it is likely that both ways are used. The first one when the problem domain is new and the second one when experience has been gained and an explicit or implicit understanding of the domain and of the way it is "segmented" into regions has been developed.

It is of course possible to define regions and outcomes without being prompted by an unsolved problem. However, people like to work from examples and this amount to an assumed problem.

The model presented above differs very markedly from the models that underpin traditional rule-based expert systems and case-based reasoning discussed previously.

1.2 COMPARISON WITH RULE-BASED SYSTEMS

In traditional rule-based expert systems, the underlying assumption is that human beings can express rules for a domain and that the power of the inference engine comes from the firing of rules (when their conditions are met) which then change the variables in the system. This leads to new conditions becoming true and new rules being fired, etc. (forward-chaining). While backward chaining operates somewhat differently, the same model is used. The features of rule-based systems are:

1. Knowledge expressed as rules is global in nature. That is, subject to the conditions being met, the rules apply to the whole of the domain space (across all combination of variables, all the values for these variables, at all times).

2. Knowledge as rules does not separate the knowledge from its domain of validity (the validity of a rule is limited only by the conditions set for the firing of the rule)

3. The inference process is global in nature. That is, it applies to the whole of the domain space (or specified subspace in object oriented systems). The inference process relies on the global nature of the rules in the system (if they were not global, the inferencing process could not operate).

4. The context is not explicit. This means that either there is an assumed implicit context or that rules are to be valid for any context. This causes problems as a) not everybody may have knowledge of, and agree with, the implicit context and b) the implicit context can change without warning. Indeed this happens every time a new variable is added to the system. In both cases the validity of the rules is put into questions.

5. Features 1, 2 and 3 contribute to the problems experienced in the verification and validation of these systems (which we can describe as the system reaching an unanticipated state, perhaps from the unanticipated firing of some rules, which in turn can lead to more "inappropriate" rules being fired, etc.). This is because the inference process is designed to take account of all rules in the whole of the

domain space. Even a relatively small application can have a very large number of conditions and possibilities, often too large for a human being to easily visualise and understand. It appears that humans cannot generate rules with sufficient accuracy when the domain of application has many variables and when the inference process is global in nature.

Traditional rule-based expert systems implicitly define knowledge as a set of incomplete items (rules) that define part of a context and part of the consequences associated with that part of the context. Each item is an incomplete description of a global context and a partial consequence associated with the (incomplete) context.

Compton & al, 1993, have proposed "ripple down rules" as a rule-based technology that addresses some of the problems of traditional rule-based systems. In ripple down rules systems, when a conclusion of the system is judged inadequate by the domain expert, he/she can add a rule that corrects the inferencing process just before it reaches the wrong conclusion. The system is modified to produces a different, correct, answer. The domain expert looks for a factor in the new problem or case that is different from the previous cases and then uses this factor in the rule. Knowledge acquisition is incremental. The domain expert certifies new knowledge by certifying the new rule at the time of creation. Certification is however not complete. A new case, different from the one that prompted the rule change, can follow the same path in the knowledge base (because the rest of the knowledge base may not recognise this new case as being different from the one that prompted the rule change). The answer produced by the system may therefore be inappropriate to the new case. The underlying cause of the problem is that the domain expert certifies each new rule and its conclusion with respect to the case that followed the path. As other different cases requiring different answers may follow the same path, the certification process is not truly valid.

It is clear from the above that "ripple down rules" systems are for domain experts. Users who are not domain experts may not recognise that a conclusion is inadequate and wrongly accept it as valid. As the system is being used, its knowledge base becomes more extensive and it is conceivable that it may be comprehensive enough with respect to the domain of use for non domain experts to use it.

1.3 COMPARISON WITH CASE-BASED REASONING

CBR systems rely on the episodic memory and a reasoning model based on past experiences. That is, old cases and their answers are the starting point for assessing new cases, which are then modified to suit the new problem and arrive at an appropriate answer. Unless the new case is equivalent to an old episode, the same answer may not be useable. This is left to the user who has to determine, at the time the problem is faced, a) whether the retrieved case is similar or similar enough to the new case in some essential features, b) if not, how to modify the retrieved case, c) how to test the new case (the modified retrieved case), and d) how to describe the new case with indexes for storage in memory. CBR also typically applies to areas where the problem is unstructured and has ill-defined concepts (Kolodner, 1993).

CBR does not guarantee that cases with similar indexes are in fact similar as the objective of indexing is to separate the cases from one another, based on the cases seen (or experience gained) so far. Some new case may have the same indexes and still require a different outcome.

2. THE SSPG MODEL - FEATURES AND BENEFITS

2.1 GENERAL DESCRIPTION

The SSPG strategy is a local strategy in an explicit global context for the problem and solution spaces. With respect to Figure 3 in the previous Section, a domain expert is responsible for specifying the problem and solution contexts and, within this global context, for answering the "precise problem" and for defining the "local region" when appropriate (that is, the region of the problem space and context where the same answer, taking account of its context, is deemed by the expert to hold). A knowledge item in the SSPG model consists of a region and its associated answer or outcome, both expressed in their explicit source and destination contexts respectively.

A useable system requires the definition of several, sometimes many, regions and associated answers. These regions are typically defined when a problem is encountered for which there is no existing answer in the system, that is, the problem does not fall within an existing region. Alternatively, the domain expert can decide to define regions in the problem space based on hypothetical problems.

2.1.1 Knowledge in the SSPG model

SSPG assumes that knowledge has no global validity. On the contrary, SSPG accepts as a fundamental principle that knowledge is inseparable from its context and domain of applicability within that context. Knowledge and its domain of applicability must be defined simultaneously, locally, by defining a region and an outcome in the application space.

Under the SSPG model, knowledge is defined as:

Knowledge consists of knowledge items expressed as mappings.

A knowledge item is made of a region in the explicit problem space which is linked to a subset of the explicit solution space.

The problem and solution spaces provide the explicit context for the knowledge.

This is illustrated in Figure 4 below which shows six regions in the source space with the precise problems (shown as • in the regions) that triggered the definition of the regions, and six corresponding outcomes (shown as • on the "outcomes" axis).

Notable features of the model are:

1. The regions can be of different shapes and sizes, as judged appropriate by the domain expert.

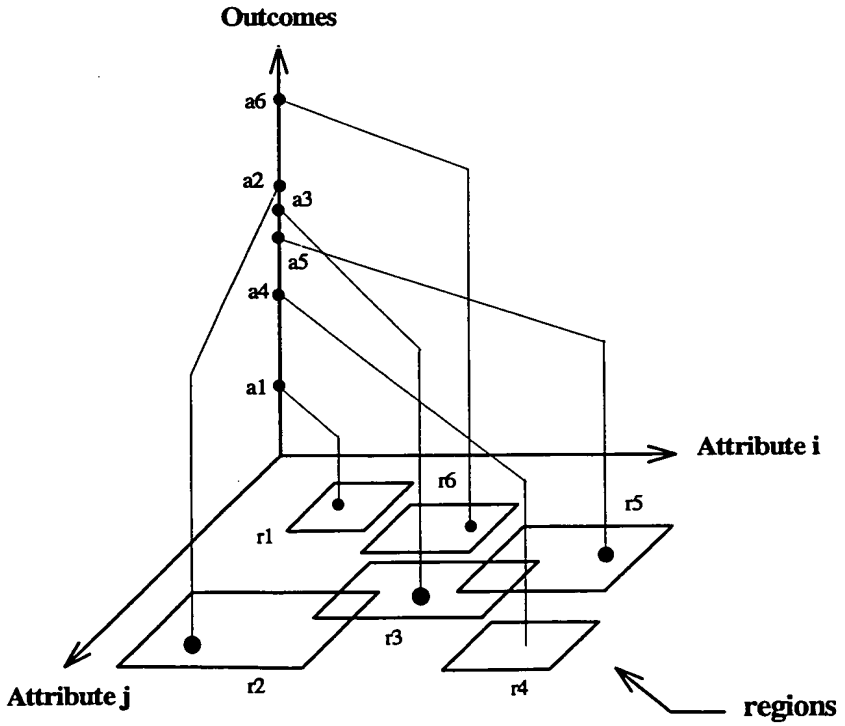
2. The precise problem does not need to be in the centre of the region. Again this is judged by the domain expert.

3. Some regions can be defined without being prompted by a precise problem. In this case there is no • in the region (region r4 in Figure 4).

4. Separate regions can have the same answer. This could mean that the domain expert took a conservative approach at the time of the region definition and certified the answer only in a small region. Later, when defining another region, perhaps prompted by a different precise problem, the expert gave the same answer but again felt confident only with respect to a small local region.

5. Regions can overlap. In this case a strategy must be developed to select the appropriate answer among the several that may be possible (discussed later).

Figure 4: Knowledge in the SSPG model



Although the answer space is represented as a single axis in Figure 4, it usually is a multidimensional space, with each outcome being a partition of the total solution space.

2.1.2 The knowledge acquisition process

The process followed by a system based on the SSPG model for knowledge acquisition is illustrated in Figure 5.

The process steps are:

1. The system invites a user to define a specific problem.

2. The system checks whether the problem falls within a region in the source space.

3. If it does, the system gets the answer corresponding to that region and gives it as answer to the new problem.

4. If the problem does not fit inside an existing region, then the system informs the user that no answer can be given at this stage.

5. The system then refers the problem to a domain expert who:

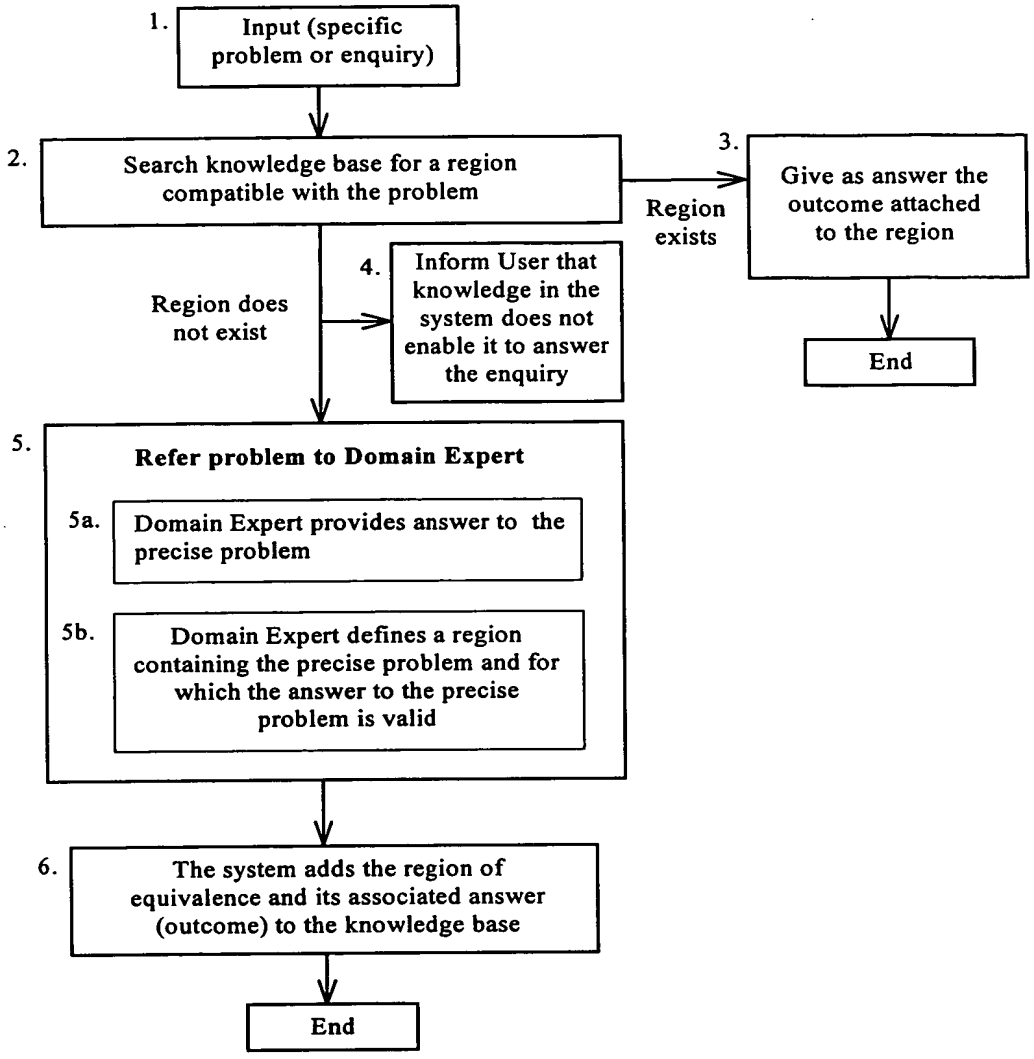
   a) specifies the answer to the problem, and

   b) defines a region for that answer.

6. The system then links the region with the answer and adds both to its knowledge base. This new knowledge now becomes available for all subsequent users of the system.

With respect to point 5 above, the domain expert vouches for the region and its outcome (the answer). In effect it verifies the knowledge (answer is the appropriate one for the problem) and validates it (the answer and the domain of equivalence address relevant situations).

Figure 5: SSPG reasoning flowchart



2.2 KNOWLEDGE REPRESENTATION FEATURES

Knowledge, in SSPG, is modelled onto an appropriate domain space (the global context) comprising a problem space and a solution space. Knowledge is represented a set of regions, each linked to an outcome. Knowledge is local in that each knowledge item (a region and its outcome) defines a localised domain of applications or types of problems (the region) and an outcome (the appropriate answer to applications or problems that fall within the region) within the global context. The domain or application space (that is, the space that contains all the knowledge items, re: all the regions and outcomes) determines the types and ranges of problems that can be addressed using the system and it is the role of domain experts to use their experience and judgment to define that space.

Because of the local property of knowledge in the SSPG model, it is possible to define a domain space with a different number of dimensions and different attributes attached to

these dimensions, for different parts of the domain space. This applies to both the problem space and the solution space.

2.3 KNOWLEDGE ACQUISITION FEATURES

With reference to Figure 5, the key features are:

1. Knowledge acquisition is incremental, usually prompted by an enquiry about a specific problem.

2. A SSPG-based system is useable even when little knowledge is included in it. As the system is being used, more knowledge is being entered and the usefulness grows.

2. Only useful knowledge (re: knowledge relevant to specific problems) is acquired. (As domain experts can define regions and outcomes without being prompted by enquiries, it is possible to include non-useful knowledge in the system, that is knowledge that will not be of use in dealing with practical problems. This is however unlikely to happen to any significant extent in practice.)

3. Knowledge is verified at acquisition time. That is, the domain expert vouches for the applicability and relevance of the knowledge to all situations and problems that fall within the region. The expert can be either very conservative and define a small region or more confident and define a larger region.

4. Knowledge is validated at acquisition time. That is, the knowledge entered is relevant to situations and problems for which the system is being built and used. This is because the definition of most of the knowledge items stored in the system is triggered by specific problems. When regions and outcomes are defined without being prompted by a specific problem, then the regions defined may have little relevance to the intended domain of applications for the system and one cannot claim that these knowledge items are validated.

5. The knowledge items have independent validity. An item can be removed without affecting the validity of the other items of the system. The rest of the system is still useable. A knowledge item can be removed and replaced by another. In a similar way, a region or its outcome can be modified without affecting the validity and the operations of the rest of the system. A SSPG-based system is robust. In contrast traditional rule-based systems are not robust in that they can be drastically affected by any changes in the rule base.

6. A stronger feature than 5 above would be that the knowledge items are independent of each other. This is however not strictly the case as new knowledge items are typically defined when new enquiries do not fall within existing regions. The definition and storage of new knowledge items is in part determined by the knowledge already existing in the system.

2.4 THE REASONING PROCESS

The reasoning process in SSPG consists of:

2.4.1 Knowledge acquisition support

When a new region and answer is being defined by a domain expert, it can happen that this region overlaps with previously defined regions. Support is required to let the domain expert know that these overlapping regions exist and to assist her/him in viewing them. The domain expert then can opt to:

a. modify the new region and/or its outcome appropriately

b. modify one or several already defined regions and/or their outcomes (if the domain expert has the authorisation to do so).

It is not necessary to remove all overlaps between regions. A case contained by several regions may have several possible or complementary answers (the outcomes of the regions containing it). It is also possible that 2 regions have the same outcome. In this situation, the answer is the same whatever region one selects.

Knowledge acquisition support is not mandatory. It is possible, for example, to simply give as answer to an enquiry at run time the outcome of the most recent region containing the enquiry (see below).

2.4.2 Solution search

Solution search processing consists of:

1. Finding the regions of equivalence that contain the enquiry. This can be done in 2 ways:
   a) without inferencing search (or non-interactive search)
   b) with inferencing search (or interactive search).
2. Selecting the region(s) most relevant to the enquiry.
3. Retrieving the outcome(s) corresponding to these regions.
4. Presenting these outcome(s) in the appropriate way to the user.

Searching for regions *without inferencing* consists of a) defining the enquiry as completely as possible and b) finding the regions that contain it.

Searching *with inferencing* is a more interactive process. It consists of asking the user only these questions about the problem that enable the system to arrive at a solution as quickly as possible. Each question elicits some features about the enquiry and directs the search towards the most relevant part of the problem space. Some features of an enquiry may not be asked as they apply to region(s) in a part of the solution space that has already been found as being irrelevant to the enquiry. Searching with inferencing can be achieved,

for example, by measuring the "regions discriminating power" of each feature in the problem space and by asking, at each step in the inferencing process, only about these features that have the highest discriminating power.

The reasoning process techniques are discussed more fully in the next Chapter.

2.5 GLOBAL FEATURES

The global features of a system based on the SSPG model, see Sections 4.1 to 4.3 above, are:

1. Knowledge is expressed as knowledge items in an explicit global context made of a problem specification context and a solution specification context. The context is accessible to users.

2. Knowledge acquisition is incremental and interactive.

3. Knowledge is certified (verified and validated) by the domain expert at acquisition time.

4. The system knows the limits of its knowledge. Its knowledge is only applicable to problems that fall within the defined regions in the problem space. If there is no region in the system containing an enquiry, this enquiry is outside the knowledge of the system.

5. The system restricts the use of its knowledge to situation it recognises (re: to problems that fall within the regions). These regions have been approved by domain experts.

6. The system informs the user when an enquiry falls outside its knowledge (re: outside the defined regions in the problem space).

7. The system asks a domain expert to increase its knowledge when an enquiry fall outside its knowledge. The system gets more knowledgeable as it is being used.

8. The system separates between a user (not a domain expert) who can get advice from the system and the domain expert who manages the system and the knowledge in it.

9. Knowledge is maintained by domain experts who can define new regions and outcomes, correct existing regions and outcomes, as required by the use of the system and by new knowledge becoming available for inclusion in the system.

Point 9 does not strictly follow for the Sections 4.1 to 4.3 as it is a matter of implementation (treated in the following chapters). It is mentioned here because it is a very useful and important complement to the other features.

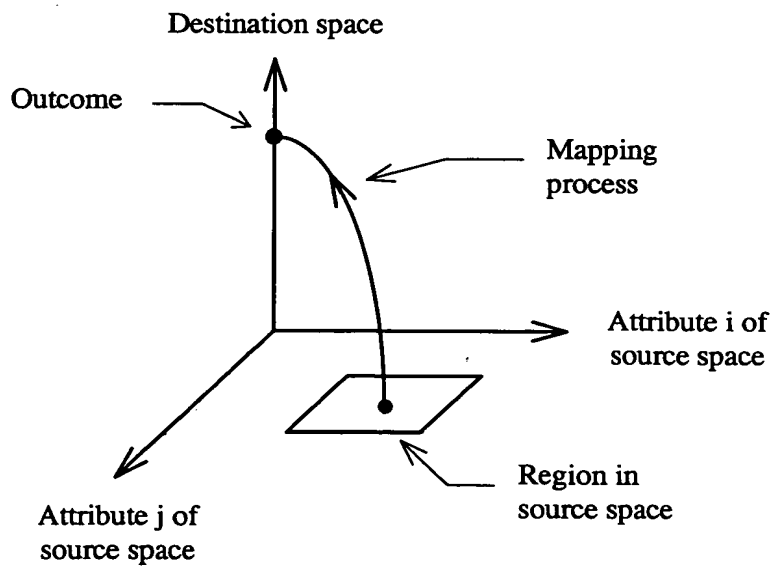
The set of features above is of paramount importance for the widespread acceptance and use of knowledge-based systems in all sectors of human activities. SSPG-based systems

can be natural extensions of human beings for the management of their knowledge. With SSPG-based systems, humans can store their knowledge, retrieve it, modify it, extend it and share it with other domain experts and with users who are not domain experts. Domain experts can use such systems as repositories of their knowledge and as tools for the development and codification of new knowledge acquired as they gain experience in their domain of expertise. SSPG-based systems can be both personal tools and systems for sharing one's knowledge with those who are less expert in the domain.

3. THE ELEMENTARY GKMS MODULE

The elementary GKMS module is the basic building block for all GKMS implementations. It comprises a source space, a destination space and a mapping. The mappings express and embody the knowledge supplied by experts to provide answers, expressed in the destination space, appropriate to the situations described in the source space. A mapping is a relationship between part of a source space onto part of a destination space, typically from a situation or group of situations in the source space onto an outcome in the destination space. In Figure 1, all the situations in the source space which are part of the region are linked to the same outcome. Both spaces are multi-dimensional.

Figure 1: Mapping from source space to destination space



The curved arrow, with its attendant source and destination spaces, its region from the source space to its outcome in the destination space represent the mapping. It specifies a region in the problem space and a way to produce an outcome when the conditions of the

problem place it within the region. The mapping process can be explanations or actions. The source and destination spaces define the context for the mapping. Mappings

3.1 EXPLANATION MAPPING

Explanation mappings are defined by their source and their destination. They are not calculated, they are stated. Explanation mappings are associated with code which enable the outcome to be displayed on a screen or printer. Explanation mappings are typically used in rule-based systems to give advice. A situation (antecedent of a rule) is linked to an outcome (consequent of that rule) which displays the advice on a screen.

3.2 ACTION MAPPING

Action mappings come in two types:

Action mappings type 1

Type 1 mappings associate a situation in the source space to actions expressed in the destination space. The destination space, instead of being explanations as in explanation mappings is made of instructions to be carried out by the some agents.

Action mappings type 2

Type 2 mappings are specified by a function or module that can be calculated, using the values of the source attributes that define the situation as parameters. The result of type 2 mappings are attributes which can take values. Programming can be viewed as the calculation of action mappings, one after another.

Action mapping examples:

- A workflow module in which the action to be taken by the system (outcome) are predicated by a situation described in the source space (type 1).
- An equation with variables and constants that are part of the source and destination spaces (type 2).
- A procedure or function (algorithm) (type 2).

The regions in the source space can be small or large. Regions can overlap and have sub-regions. The outcome can be a point or a region in the destination space. The mapping process described above is a generic process that covers all that can be expressed using logical and mathematical expressions.

3.3 THE ELEMENTARY GKMS MODULE

The elementary GKMS module comprises:

- A source space or context.
- A destination space or context.

- At least one mapping that can be action or explanation (that is, at least one knowledge item).

3.3.1 The source space

The source space is made of objects (or attributes) that enable the user (expert) to define situations. The source space typically contains attributes that define the external world. However it can also contain attributes that define the internal state of the system. When it is the case, mappings can be used to take actions based on the state of the system, that is, mappings can be used to control the behaviour of the system (this point is developed further in Section 3.11).

The destination space can also be part of the source space. The two spaces can overlap.

3.3.2 The destination space

The destination space is made of attributes that describe actions, events, statements about the external world. It specifies the actions that can be taken to modify the external world or the internal state of the system.

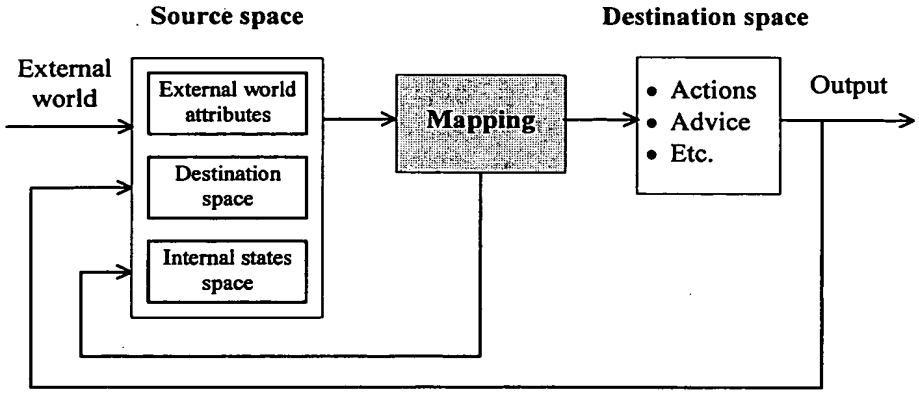
3.3.3 The GKMS model

The GKMS model provides a framework for expressing knowledge. Knowledge is expressed as mappings from a source space to a destination space. The source and destination spaces provide the context for the knowledge. As explained in Section 2, making the context for knowledge is essential for assessing its value correctly and for using it correctly.

The source and destination spaces, together with the type of mappings allowed by the system (type 1 or type 2) form the 'domain of discourse'. It defines the environment in which mappings take place; that is in which knowledge is expressed.

Figure 2a illustrates the GKMS model with the source space comprising (subsets of) the destination space and internal state of the system. The input into the source space comprises elements of the external world, of the destination space and of the internal state. It is also possible to have the output included which becomes a subset of the destination space component of the input.

Figure 2: Elementary GKMS module design



Time can be an object both in the source and destination spaces.

3.3.4 The elementary GKMS module

The elementary GKMS module is the working implementation of the GKMS model. This module enables experts to define the source and destination contexts and mappings between these contexts. It also enables users (who are not experts) to access or use the expertise which is embodied in the module and its mappings.

The elementary GKMS module supports the following functions:

- A source and destination spaces editing sub-module
- A explanation mapping editing sub-module
- An action mapping editing sub-module
- A knowledge base
- A query definition sub-module
- A processing sub-module
- A system behaviour sub-module

The GKMS module has 2 types of users:

a) The expert who is responsible for defining the source and destination spaces, and for entering and managing the mappings (knowledge) in the module. He/she is accountable for the accuracy and currency of the mappings (knowledge).

b) The non-expert (or user) who defines (interactively with the system) enquiries with a view of getting knowledge (advice, recommendations, etc) from the system, the results of actions carried out by the system.

3.4 SOURCE AND DESTINATION CONTEXT EDITING SUB-MODULE

This sub-module enables authorised users to define and modify the destination and source spaces or contexts. These spaces are constructed out of attributes or objects. Contexts are defined as a hierarchy of attributes which are categorised in groups of folders. This is similar to the file system on a personal computer, with enables users to organise their files or documents in folders and subfolders. Context editing enables experts to define attributes or objects used to describe the conditions for a mapping to take place (source context) and the range of possible outcomes of mappings (destination context). It also enables the experts and to organise these attributes in folders and subfolders. As explained in the next Section, the source and destination contexts represent the domain in which mappings (or knowledge) are expressed. As such experts need to include in the source and destination contexts all the relevant attributes which define the domain of applicability of the mappings.

Note that the hierarchical structure of the contexts is not used in the processing (see Section 3.8).

Table 2 shows these objects with their properties and types. The dimensions of the source and destination spaces are equal to the number of objects or attributes in these spaces.

Table 2: Context objects and their properties

Description	Properties
• Can act as a folder and/or attribute • As folder: container for other objects, identifies a class • As attribute: specifies part of a condition, an explanation or an action	• Object number • Title • Object type (folder or attribute) • Has members (objects) • Some or none of its objects specified • Multimedia explanation (each attribute can be explained by the expert) • Date created • Author • Status: active or deactivated (a deactivated attribute is one that is no longer part of the current context and cannot be used) • Mappings it belongs to • When attached to a mapping: - complete source and destination contexts available when mapping was defined - importance level in determining whether the mapping can fire - confidence level that the attribute belongs to the outcome • Attribute type (explanation or action) • Explanation: list, logical, numeric, text, constant and their allowed values or ranges of values • Relationships between values: any, all, not • Action: software, its inputs and outputs

Table 3: Context object definition

Properties (fields)	Explanation
Title	
Explanation	• Brief multimedia description of the knowledge or action embodied in the mapping
Number	• Unique number
Date created	
Author	
Status (active or inactive)	• A deactivated mapping is one that is still in the elementary GKMS module but that cannot fire
Knowledge items it belongs to: - to regions for source attributes - to outcomes for destination attributes	• Knowledge items identified by their numbers

The implementation of this sub-module can be done easily using modern software development tools.

3.5 MAPPING EDITING

Mapping editing enables experts to define mappings which embody knowledge. Mapping definition takes place in the source and destination contexts.

Experts define a region in the source space, and attach it to an outcome in the destination space (the outcome can also be a region). This sub-module presents the expert with the source context which allows the selection of attributes which belong to a region and the specification, for each attribute, of the range of values which mark the borders of the region in each dimension (each attribute is a dimension). The region specifies the conditions which determine whether a mapping can fire. The process is similar for the destination space. The two regions are then linked and identified as a mapping. Each item is an object in the GKMS module.

The two contexts (source and destination) are explicit and, jointly, form the domain (domain of discourse or expertise) for the mapping. Mappings represent knowledge with respect to their explicit domain of discourse. This point is very important as any form of knowledge is context dependent; that is, it is associated to an explicit domain of discourse. When a mapping is dissociated from its domain of discourse (by expressing only its region and outcome without specifying the complete source and destination contexts for example) then it ceases to represent precise and reliably useful knowledge. As source and destination contexts can vary, a module can contain mappings each defined with respect to a different domain of discourse.

The elementary GKMS module enables experts to define knowledge (in the form of explanation or action mappings) which is explicitly context dependent.

The regions in the source context determines when a mapping can fire. This process is 'location independent'; that is, it is independent from where the mapping is located in a system. This can be contrasted to usual programs where the knowledge about the processing is located in the code itself and the operations depend on the location of the code in the program. Location independence is seen as a major advantage in that it frees developers from the issue of location. Processing behaviour depends only on the conditions for a mapping to take place, expressed as a region in the source space. There is however an additional load put upon the system that now has to find which mapping applies next, a requirement that is taken care of in normal programming by the location of the code.

3.5.1 Explanation mapping editing

This sub-module enables experts to define and edit mappings (or knowledge items) of the explanation type.

Table 4: Object range specification for source and destination attributes in explanation mappings

Space attribute	Source and destination spaces range specification
List variable	• Select more than one item in its list as compatible items • All items except those selected
Logic variable	• Specify 'true' or 'false' • Specify 'does not matter'
Numeric variable	• Specify compatible range, or • Specify complement of compatible range • Specify 'does not matter'
Date/time variable	• Specify compatible date or range of dates • Specify complement of compatible date or range of dates • Specify 'does not matter'
Text variable	• Specify exact or partial compatible match, or • Specify complement of exact or partial compatible match • Specify 'does not matter'

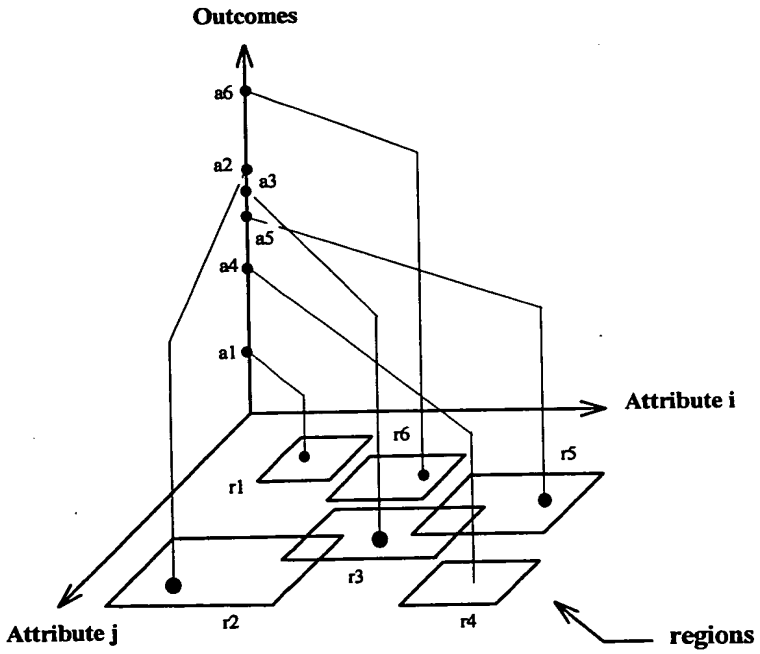
The specification 'does not matter' can be either explicit or implicit. In the first case, the expert has to specify the range (even 'does not matter') for each object (in the region) in the source space. In the second case, the objects are assumed to have the default value 'does not matter' unless a range or value is specified.

When an object in the destination space is attached to a region (that is the object becomes part of the outcome of an inference process) then it automatically becomes a member of the source space, preferably into a folder labelled 'inferred objects'. An 'inferred object' can also be identified with an icon or a colour and be located anywhere in the source space list. When an inferred object is attached to a region in the source space, then the source and destination spaces are said to be overlapping. When an inferred object is attached to a region, the region is then labelled as 'inferred region' (an inferred region has

at least one inferred object). Objects and regions that are not inferred are also described as primary objects and regions.

Figure 3 illustrates explanation items in the source-destination context. The • in all regions except r4 indicate that an enquiry was presented for which there was no answer. The system then presented the enquiry to an expert who attached it to an outcome. The expert defined a region around the enquiry so that all future enquiries falling inside the region produce the same outcome. This last steps add usefulness to the knowledge stored in the System as the item becomes applicable to a range of situations rather than to one situation only.

Figure 3: Knowledge items in the source-destination spaces



The explanation mapping is specified by the properties listed in Table 5.

Table 5: Explanation mapping definition

Properties (fields)	Explanation
Title	
Summary	• Brief multimedia description of the knowledge or action embodied in the mapping
Number	• Unique number
Date created	
Author	

Status (active or inactive)	• A deactivated mapping is one that is still in the elementary GKMS module but that cannot fire
Attributes in the source context	• List of attributes (identified by their numbers)
Attributes in the destination context	• List of attributes (identified by their numbers)
Source attributes in the region and their ranges	• Specifies the region
Destination attributes in the outcome and their ranges	• Specifies the outcome
Status	• Has fired / has not fired
Belongs to either the source or destination context	• The mapping is itself an object that can be used to enrich the context
Reliability level	• Describes the importance or reliability of the mapping

3.5.2 Action mapping editing

This sub-module enables experts to define and edit mappings (or knowledge items) of the action type.

Table 6: Object range specification for source and destination attributes in action mappings

Space attribute	Source and destination spaces range specification
List variable	• Select more than one item in its list as compatible items • All items except those selected
Logic variable	• Specify 'true' or 'false' • Specify 'does not matter'
Numeric variable	• Specify compatible range, or • Specify complement of compatible range • Specify 'does not matter'
Date/time variable	• Specify compatible date or range of dates • Specify complement of compatible date or range of dates • Specify 'does not matter'
Text variable	• Specify exact or partial compatible match, or • Specify complement of exact or partial compatible match • Specify 'does not matter'

Action object	• Specify compatible states (specify input and/or output states or ranges) • Specify complement of compatible states • Specify 'does not matter' • Automated activation of object or user controlled activation (at knowledge access time, see Section 3.4.4)

The action mapping is specified by its properties as listed in Table 5.

In an action mapping, the expert specifies the objects (typically numeric and logical) that are the input to the mapping and then defines the mapping. The output is computed, not specified in a static way as for explanation mappings. The mapping is defined as follows:

- The expert selects the source space objects that go into the mapping computation.
- The expert selects or define an operation or computation. All the system needs to support initially are the elementary mathematical and logical operations. Do Loops (Do While, Do Until, etc) may also be included as part of the 'primitive' operations. However, if one deals with low level programming, the primitive set may include the operations that are carried out on the registers in a microprocessor.
- The outcome is the result of the computation. It automatically becomes part of the destination space and part of the source space (preferably into a folder labelled as 'calculated outcomes').

The last point ensures that action mappings can be chained to produce arbitrary complex calculations. It means that the source and destination spaces overlap.

There is a similarity between the software objects mentioned in 3.5 and the action mappings described here. An action mapping is defined in the GKMS environment whereas a software object has been defined elsewhere (it could be part of a library of procedures for example).

If an action object is part of the destination space, then the region in the source space specifies the conditions when the software object becomes (part of) the outcome. At access or run time an action object as part of the outcome can mean:

- Modify the object and put it in the state specified in the outcome.
- Activate the object automatically (automated activation, using the input state the object is in).
- Give user option to activate the object (user controlled activation, using the input state the object is in).

3.6 THE COMPOSITE GKMS MODULE

The composite GKMS module is a collection of elementary GKMS modules. It comprises the same elements as the elementary GKMS module (Section 3.). These elements in the

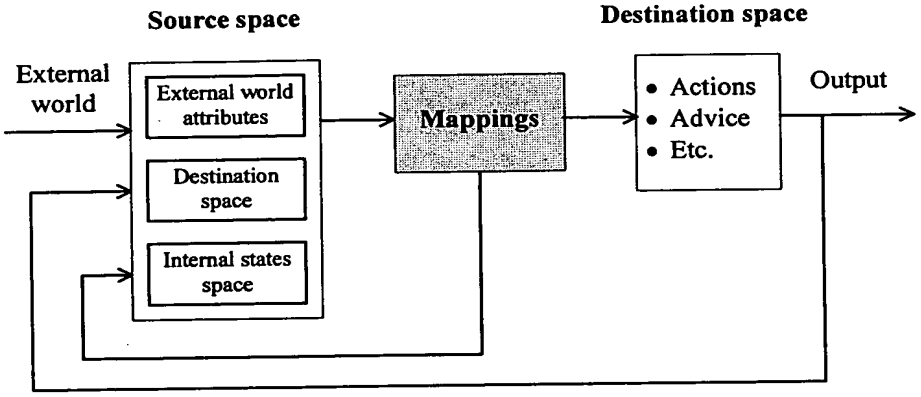
composite module are expressed as the union of the corresponding elements in the elementary module.

Table 6: Elements in the composite GKMS module

Element	Explanation
Source context	Union of the contexts in the elementary modules
Destination context	Union of the contexts in the elementary modules
Mappings	Union of the mappings in the elementary modules

It is frequent that the elementary modules which comprise the composite GKMS module have the same source and destination contexts.

Figure 4: Composite GKMS module design



A composite GKMS module made of elementary modules with overlapping contexts is referred to a knowledge base.

3.6.1 Concatenation of knowledge bases

Composite GKMS modules can be concatenated or grouped to build larger knowledge bases. When composite modules have overlapping contexts they are described as overlapping knowledge bases. When they have non-overlapping contexts they are described as disjointed or independent knowledge bases.

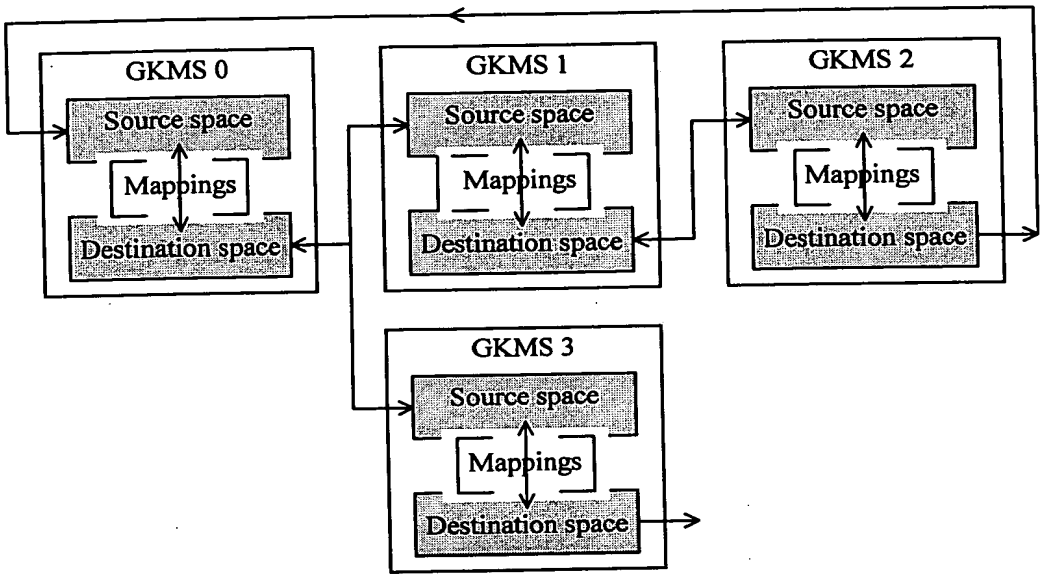
When independent knowledge bases need to be concatenated, then relationships between the knowledge bases need to be made explicit. For example (part of) the destination context of one composite module can become (part of) the source space of another module. Another possibility is that both the source and destination of the two composite modules overlap.

In some situations the contexts of the knowledge bases are disjoint. In this case, the link between the two knowledge bases is done by defining another module or knowledge base

which links and explains (some elements of) the contexts of the independent knowledge bases.

For example (see Figure 5), the destination space of one module becomes the source space (or part of the source space) of another GKMS module. Module 0 is a source GKMS to the destination modules 1 and 3 and module 2 is a source GKMS to module 0. The networking of GKMS modules gives flexibility for knowledge management and accountability. For example, module 0 could deal with the hardware aspects of a microprocessor device, module 1 with the low level software, module 2 with application software, etc. Each module can have different domain experts who are responsible for the quality and currency of the knowledge for their module only.

Figure 5: Network of GKMS knowledge bases



Knowledge certification in networked modules can be either local or global. Local certification is when each AKMS module is certified independently of the other modules. A change to the destination space of a source module (say AKMS 0 in Figure 5) does not affect the certification of the destination module (AKMS 1 and 3 in Figure 5). Global certification requires all destination modules to be re-certified when a change takes place in the destination space of a source AKMS.

At the implementation level, the expert is presented with a diagram (graphics) similar to that in Figure 5, without the connecting arrows. The expert then can add the connecting arrows to define the links between the AKMS modules. By clicking on a module, the expert is taken to the module itself and has access to all the functionality of the elementary AKMS module. For processing, the structure above is collapsed onto a single two layer structure (a single source space and a single destination space) as explained in Section 3.10.8.

3.7 QUERY DEFINITION SUB-MODULE

This sub-module enables a user to define a query in the source space, or a process to perform the role of a user. In effect, the user defines a situation for the knowledge processing module to act on. Each object in the source space can take three values in an enquiry: i) specified, ii) 'don't know' and iii) 'unspecified'. The query definition can be one of two modes:

3.7.1 Non-interactive enquiry

The GKMS presents the user with the complete source space. The user then specifies a situation and the GKMS looks for regions compatible with the query. If there are compatible regions, then their outcomes are presented to the user. If there isn't any compatible region, the user is informed. In this case, the query can be transmitted to

3.7.2 Interactive enquiry

The GKMS presents the user with one or a few questions at a time for the user to answer. Once it is done, the GKMS processes the answer(s) and determines the next best question(s) to ask. The best questions are those that lead to the mappings (re: regions) compatible with the query with as few questions as possible. When the system has identified a compatible region, then it presents its outcome to the user. If the system cannot find a compatible region, it informs the user.

3.7.3 Hybrid enquiry

The GKMS presents the user with several questions grouped logically (re: addressing a single issue). Once the user has answered these questions, the system presents the next group of questions, or single question, as the case may be.

3.8 KNOWLEDGE PROCESSING

Knowledge processing is the process used by GKMS to identify whether there are regions that are compatible with the enquiry. Knowledge processing starts with a list of candidate regions or mappings (initially all the regions in the knowledge base) and on the basis of the objects specified in the enquiry, that is the values of the dimensions in the source space, determines:

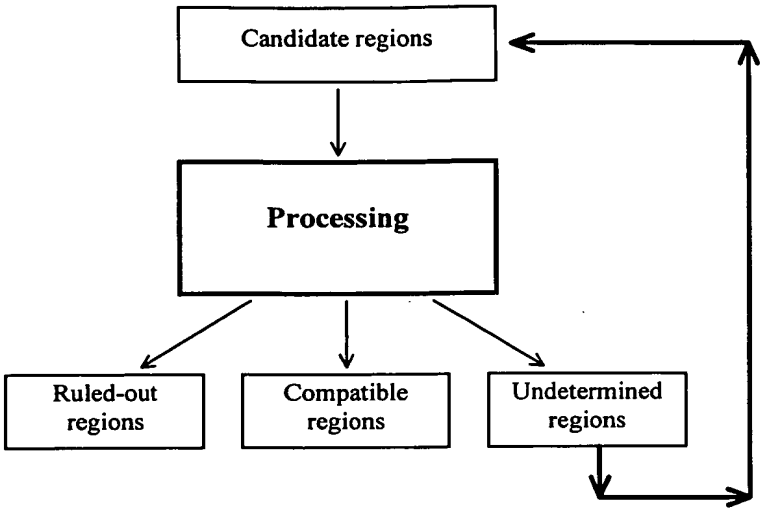
a) the regions that are ruled out by the enquiry;

b) the regions that are compatible with the enquiry;

c) the regions that are undetermined (that is, the regions for which one cannot say whether they are compatible or ruled out because some questions have not yet been asked).

The process is described below.

Figure 6: Knowledge processing



The processing stops when a) there are no more candidate regions or mappings, b) there are no more undetermined regions or mappings, or c) there are no more objects whose values have not been determined.

Processing can stops with several outcomes:

a) There is at least one or more regions compatible with the enquiry

b) There is no region compatible with the enquiry.

In the second case, the user is informed and the query is stored and presented to an expert who then defines a mapping (action or explanation, with a region and an outcome), with the region containing the enquiry.

3.8.1 Location independence

The process described above is 'location independent'. The knowledge that drives the processing resides in the mappings. This can be contrasted to usual programs where the knowledge about the processing is located in the code itself and the operations depend on the location of the code in the program. Location independence is seen as a major advantage in that it frees developers from the issue of location. Processing behaviour depends only on the conditions for a mapping to take place, expressed as a region in the source space. There is however an additional load put upon the system that now has to find which mapping applies next, a requirement that is taken care of in normal programming by the location of the code.

Location independence is seen as particularly advantageous in 'inference chaining', Section 3.10.7.

3.8.2 Non-interactive processing

The GKMS takes as input the query defined by the user and looks for regions in the source space that are compatible with the query. A region is compatible with the query if the region "contains" the query. This means that the value of each object in the region contains the value of the object in the query (see Table 7 below for the meaning of "contain").

This applies whether the 'does not matter' is explicit or implicit in the regions (see Section 3.5.1) but the system must check all objects in the source space and the dimension (number of space dimensions required to specify the region) of each region is equal to the dimension of the source space. This can be a significant computing overhead when many dimensions have values 'do not matter' in the specification of the region.

It is possible to store regions with less memory space if the dimensions that do not matter are not included explicitly in the regions. Only the dimensions with specified values are explicitly identified. At processing, unspecified dimensions of regions are taken to have the value 'does not matter'. Alternatively, and it is equivalent, the system can check that all dimensions specified in the regions are contained by the objects specified in the query (in this case a region contains a query if the query contains the compacted form of the region). This is computationally advantageous.

Table 7: Meaning or "contain"

Space attribute	Meaning of 'contain' (in A contains B)
List variable	• List parameters in B are present in A • Value of A 'does not matter'
Logic variable	• Value of A is equal to value of B • Value of A 'does not matter'
Numeric variable	• Range of A contains the range of B • Value of A 'does not matter'
Date/time variable	• Range of A contains the range of B • Value of A 'does not matter'
Text variable	• String A contains string B, or • Value of A 'does not matter'
Software object	• Input and/or output ranges of A contain the corresponding ranges of B • Input and/or output states 'do not matter'

Practically, users only define the values of the objects that are deemed relevant to the enquiry. Some objects may be left unspecified but this could mean that i) the user does not know the value of these objects in the problem/situation at hand or ii) the user erroneously considers these objects as being irrelevant to the enquiry. Two processing approaches can be taken:

a) 'don't knows' are treated as equal to 'unspecified' and vice versa. In this case GKMS considers all the objects or dimensions in the source space for processing.

The system is able to determine which regions are compatible with the enquiry. All the other regions are deemed incompatible with the enquiry.

b) 'don't knows' are different from 'unspecified'. In this case the GKMS only considers the objects whose values have been specified for processing. On the basis of this limited number of dimensions, the system (GKMS) determines which regions are compatible, which are ruled out and which are still candidates:

- ruled out regions are those that are not compatible with the enquiry on the basis of the specified objects in the enquiry.
- compatible regions are those that are compatible with the enquiry on the basis of the specified objects in the enquiry and for which all the unspecified objects or dimensions have values (in the region) equal to 'does not matter'.
- candidate regions are those that are compatible with the enquiry on the basis of the specified objects in the enquiry and for which some of the unspecified objects or dimensions have values (in the region) that are not 'does not matter'.

When some candidate regions still exist, the system moves to an interactive processing mode to determine whether some of these candidate regions may be compatible with a more precise enquiry (see the next three sections below: Interactive processing, Hybrid processing and Sub-regions).

3.8.3 Interactive processing

The system calculates the 'discriminating power' of each object (or dimension) in the source space the value of which has not yet been specified. The discriminating power can be calculated in 2 ways:

a)
- Calculate the number of regions (or mappings) for which the value of the object does matter.
- The discriminating power is the number divided by the number of regions in the candidate list.

  (this assumes that the number of regions for each of the legal values of the object is approximately the same).

b)
- Calculate the number of regions (or mappings) for which the value of the object does matter, for each possible value of the object (the range of values for each object is divided in segments).
- The discriminating power is the average number of regions per segment, multiplied by the number of segments and divided by the number of regions in the candidate list. This number is weighted with (in the first instance, divided by) its variance (the lower the variance the higher the discriminating power).

Table 8: Segment identification in the calculation of the discriminating power

Space attribute	Segment identification
List variable	• Each list parameter (item on the list) is a segment • Number of segments = number of list parameters
Logic variable	• The segments are: 'True', 'False' and 'Does not matter' • 3 segments
Numeric variable	• The legal range of the variable is divided in non-overlapping segments • The segments boundaries are defined by the end-points of the ranges for each of the regions (for which the variable matters) • A region range can be made of several segments • A segment can belong to several regions
Date/time variable	• As for numeric variables
Text variable	• The segments are: 'Contains', 'Does not contain' and 'Does not matter' • Alternatively, a text variable can be treated as a list variable, with the list parameters being the string characters
Software object	• As for numeric variables (when 1 variable only) • Complicated when there is more than 1 variable

[Note: The GKMS discriminating power is calculated in a way that has parallel with Case Based Reasoning (CBR) discriminating power. In CBR the discriminating power is calculated for each index by counting the number of cases the index appears in, divided by the total number of cases. However, in CBR, the meaning to be attached to (re: what to do with) the answer received once the question about the highest index is asked is unclear as it has to be considered in terms of a similarity measure of candidate cases with the enquiry. The situation is clearer and more simple with the GKMS. A question enables the system to determine whether a region is ruled out or compatible with an enquiry or not!]

Once the discriminating power of each object is calculated, the system asks the question with the highest discriminating power (if several objects have the same discriminating power, then the system either presents all the related questions or selects one of them only for presentation to the user). Once the user has supplied the answer related to the object with the highest discriminating power, the system uses the answer to remove from the list of candidate all the knowledge items that are ruled out by the answer. The process described above is then repeated, that is, the discriminating power of each non-specified object or dimensions in the remaining candidates is then calculated, etc. This is carried out until there are no more candidate regions, no more undetermined regions or no more dimensions for which the values have not been ascertained.

Alternative way of calculating the discriminating power

3.8.4 Forward and backward chaining

Interactive processing can be a) forward chaining or b) backward chaining.

Forward chaining processing selects, via the interactive question/answer session, the next best question that will identify the region, if any, that contains the query. Backward chaining processing selects, via the interactive question/answer session, the next best question that will identify the region, if any, that is compatible with the desired outcome. With backward chaining, users can define a desired outcome by selecting from the objects in the destination space and then find out whether their situation applies. The desired outcome may require the situation to be described by more than one region. In backward chaining the system calculates the discriminating power of source dimensions as follows (see Section 3.8.3 for comparison):

a)
- Calculate the number of knowledge items for which the value of the objects in the desired outcome do matter.
- The discriminating power is the number divided by the total number of outcomes

  (this assumes that the number of outcomes for each of the legal values of the object is approximately the same).

or:

b)
- Calculate the number of knowledge items for each possible value of the objects in the desired outcome (each object value can be divided in segments).
- The discriminating power is the average number of regions per segment weighted with (in the first instance, divided by) its variance (the lower the variance the higher the discriminating power).

Non-interactive processing can also be backward chaining (forward chaining was dealt with in Section 3.8.3). In this case users define a desired outcome and the systems informs them of the regions (if any) that are compatible with the stated outcome. The system does that by finding which existing outcomes in the knowledge base contain the target outcome and then selects the one(s) with the number of stated dimensions nearest to the number of stated dimensions in the target outcome. If no region is compatible, the system can calculate which dimension(s) of the outcome would need to be changed for the modified outcome to be compatible with a region, by the system finding which outcomes in the knowledge base has the nearest number of compatible dimensions (either too many or too few) with the desired outcome. The system then presents these outcomes.

3.8.5 Hybrid processing

Hybrid processing takes place when a user starts by defining an enquiry in a non-interactive way, using a subset of all the dimensions in the source space (either the system presents only a subset or the user makes use of only a subset). The system then selects the knowledge items compatible with the enquiry and, if there are more than one, guides the user to the most appropriate knowledge item, if any, using interactive processing.

3.8.6 Sub-regions

It can happen that the system identifies one or several regions that are compatible with the enquiry. The system may also detect that there are still regions that are candidates with the enquiry (that is, knowledge items that have not been eliminated on the basis of the information supplied by the user, either by specification of an enquiry in a non-interactive way or by answering the questions asked by the system so far). In this case, the system move into an non-interactive processing mode to determine which is the best next question to ask in order to identify the most appropriate knowledge item. This situation happens when not all dimensions have been specified as explained in Sections 3.8.1 and 3.8.2 and also when a region has sub-regions. While the region is compatible with the enquiry defined so far, a more precise answer may be given if the problem situation can be determined to in fact belong to one of the sub-regions. The system identifies sub-regions by checking whether a region contains another (see Table 4 for the meaning of 'contain').

3.8.7 Inference chaining

Inference chaining uses inferred objects (inferred objects are a members of the destination space) attached to primary regions r(i) in the source space) which are themselves part of inferred regions, say r(j), in the source space. When the value of an inferred object is 'true' or valid, then the inferred region r(j) it is part of in the source space can become part of an answer to an enquiry if the other required conditions are satisfied. The outcome attached to inferred region r(j) is then the (or a possible) answer to the enquiry. This chaining effect can be repeated as many times as an expert wishes.

Inference chaining applies to explanation and action mappings.

As explained in Section 3.10.1, the property of 'location independence' of GKMS processing is a major advantage for software developers in that it frees them from the need to consider not only the code but also its location to determine its impact on the behaviour of the system.

3.8.8 Collapsing regions

Inference chaining is based on objects whose value is the outcome of an inference. A question arises regarding their discriminating power (see above). The discriminating power of inferred objects does not need to be calculated. Only the discriminating power of non-inferred objects is calculated. An outcome attached to a region which has inferred objects is selected when its 'collapsed equivalent' region is compatible with the enquiry. The 'collapsed equivalent' region is the region obtained in the source space when all the inferred objects are replaced by the objects in the region to which they (the inferred objects) belong. By collapsing the regions, one can also determine whether the inference chaining is consistent. That is, whether some objects in the inferred regions are required to take two different values for the chain to be inferable to the end (re: whether a primary region requires a certain value to an inferred object to be activated which in is a member of another region that requires the same object to take a different, incompatible value for the inference chain to continue).

3.9 SYSTEM BEHAVIOUR

The system behaviour is specified by considering the state the system is in and in taking actions based on this state. The state of the system can be included as part of the source space and the outcomes that manipulate the state of the system can be part of the destination space. Alternatively, the system behaviour can be specified using a separate module that has as source space a subset of the source space mentioned above and as destination space a subset of the destination space mentioned above.

The system behaviour module enables the expert to define the behaviour of the system it is attached to. It offers an alternative way to implement interactive processing (where the system decides which questions to ask next) in which the expert specifies what the system should do under defined conditions. The system behaviour module is optional.

The system behaviour module is in an elementary GKMS module which consists of the same source space as that of the system it is attached to and a different destination space. The state of the system is described as in the elementary GKMS module, using regions attached to outcomes of specified behaviours (the destination space). The destination space specifies behaviour options for the system and has the same attributes or objects as the source space but with different possible values for actions specification, and some other actions as well. Table 6 shows the destination space of a behaviour module.

Table 9: System behaviour destination space

Space attribute	Destination spaces value options
List variable	• Display variable as enquiry • Hide variable
Logic variable	• Display variable as enquiry
Numeric variable	• Display variable as enquiry • Hide variable • Specify legal range
Date/time variable	• Display variable as enquiry • Hide variable
Text variable	• Display variable as enquiry • Hide variable
Software object	• Display variable as enquiry • Hide variable • Specify state (specify input and/or output states or ranges)
Other	• Save enquiry • Activate processing (re: search for advice) • Hide all variables displayed after variable x (variables in the display can also be identified with a number) • Display another object which is not part of the problem space (text, image, video, etc) as advice of supplementary information • Abort process

3.10 KNOWLEDGE REVIEW

Experts need to be able to find out conveniently what knowledge items mention some objects. To find that out the expert defines a query which specifies objects and their values, in the source and destination spaces and instructs the system to retrieve all knowledge items that are contained by the enquiry (the unspecified objects are assumed to have the value 'does not matter'). The expert can then review, update and deactivate these knowledge items.

3.11 OVERLAPPING KNOWLEDGE ITEMS

As shown in Figure 3, knowledge items can overlap and when this occurs experts need to know. Overlap can be in the source space or in the destination space. Overlap happens when a region contains part of another region. Sub-region is a special case of overlapping regions in which one region contains the whole of another region. The meaning of 'overlap' is explained in Table 7 (see Table 4 - meaning or 'contain' - for comparison).

Table 7: Meaning or 'overlap'

Space attribute	Meaning of 'overlap (in A overlap with B)
List variable	• Some list parameters in B are present in A • Value of A or B 'does not matter'
Logic variable	• Value of A is equal to value of b • Value of A or B 'does not matter'
Numeric variable	• Range of A overlaps with the range of B • Value of A or B 'does not matter'
Date/time variable	• Range of A overlaps with the range of B • Value of A or B 'does not matter'
Text variable	• String A overlaps with string B (some characters are common) • Value of A of B 'does not matter'
Software object	• Input and/or output ranges of A overlaps with the corresponding ranges of B • Input and/or output states 'do not matter'

Overlap can be checked at any time by the user but can also be checked automatically whenever a new knowledge is defined or an existing knowledge item is modified, that is, whenever the knowledge base is modified.

3.12 KNOWLEDGE CERTIFICATION

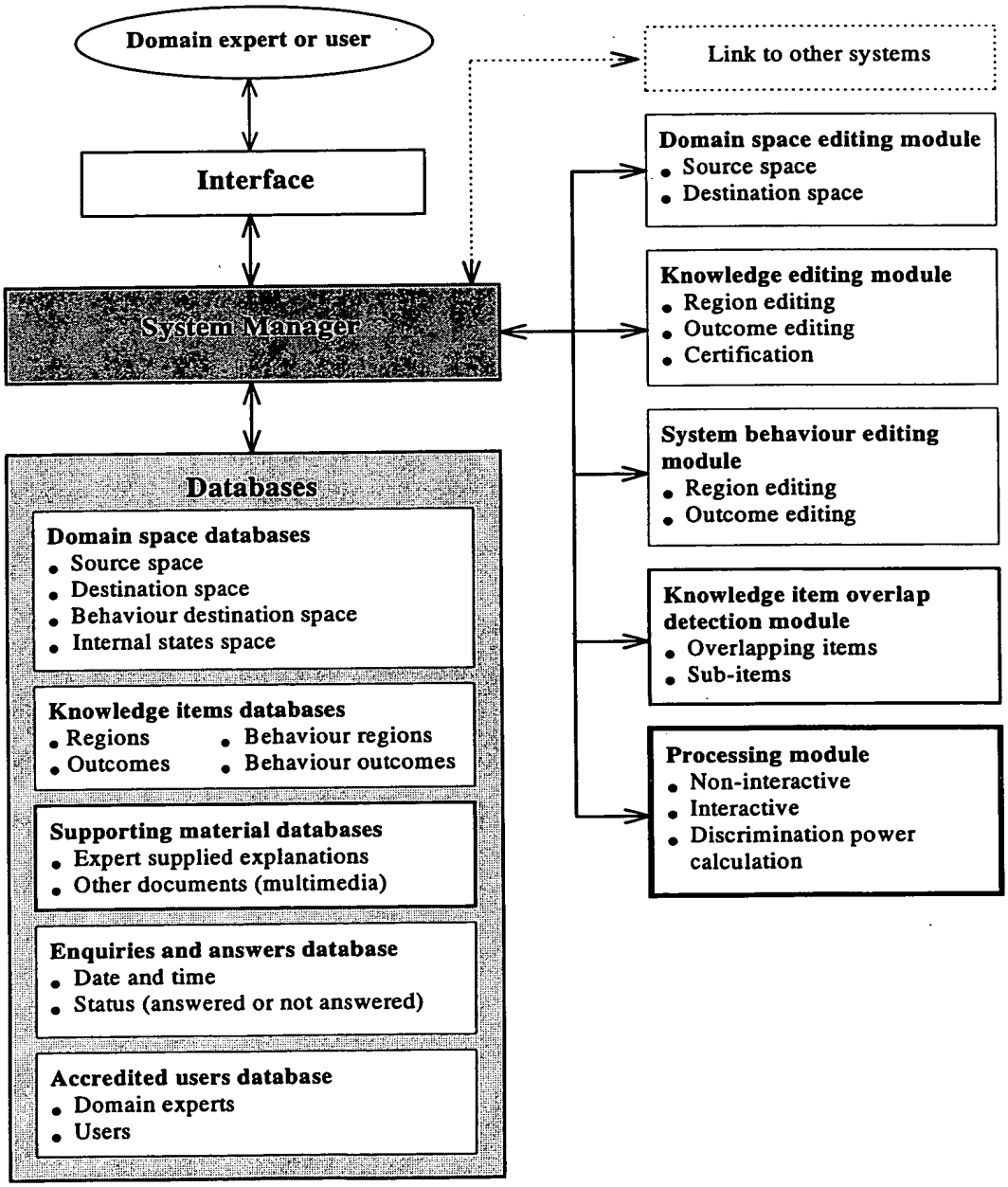
With reference to Figure 3, knowledge items are certified with respect to explicit source and destination contexts. When the context is modified, knowledge items can still be used as long as their context is made explicit to the users. Alternatively, the knowledge items

may need to be re-certified by experts. The expert may need to modify some knowledge items before re-certification.

4. THE GKMS MODULE ARCHITECTURE

The architecture is shown in Figure 5 which lists all the elements mentioned in Section 3.5 .

Figure 5: Elementary AKMS module architecture



Part 3: GKMS Implementation

1. INTRODUCTION

Practical implementation issues for GKMS are treated in this Section. The emphasis is on solutions which support the GKMS model and which are both computationally effective as well as user friendly. In all cases, the implementation is designed to enable non-computer specialists to build and use knowledge systems.

2. CONTEXT EDITING

Context editing needs to be done in two situations:

1. To define an initial domain of discourse before any knowledge items have been defined
2. To modify (usually extend) a domain of discourse to enable it to deal with an extended range of situations and problems.

2.1 INITIAL CONTEXT EDITING

Users are presented with a blank context (or a context with one or two objects in it as examples). Users can then:

- Define an object or attribute
- Edit an existing attribute
- Delete an existing attribute

The fields used to define an attribute have been defined in Part 2, Section 3.5.1 and 3.5.2. The explanation for the meaning of the attribute is a Rich Text Field which enables images, graphics, audio, etc. to be attached to the field.

2.2 MODIFICATION OF AN EXISTING CONTEXT

An existing context can be edited as explained in section 2.1. A second way of editing the context is in response to new queries which require its modification (usually extension). This second way is treated in Section 3. as it is part of knowledge acquisition in a variable context.

3. KNOWLEDGE ACQUISITION

Knowledge acquisition is also referred to as mappings definition.

3.1 ACQUISITION IN A FIXED CONTEXT

The context has been defined by domain experts to their satisfaction. That is, the domain of discourse has been set and the task is now to enter knowledge in the form of mappings.

3.1.1 Acquisition not prompted by enquiries

In this situation the expert(s) use their experience to decide, without external prompts, what mappings are needed. They then define these mappings by defining regions in the source context and outcomes in the destination context. The mapping definition process is to define a situation or problem in the source context and then to specify the outcome corresponding to the problem. An alternative way is to define a solution which is known to be useful or relevant to the domain of discourse and then to specify the region(s) which determine when this outcome is applicable. In practice the two approaches can often be merged.

Knowledge acquisition is a variant of the process described in the next Section. The process is as follows:

1. Define a region in the source context
2. Define an outcome in the destination context
3. Save knowledge item (specify a title and a summary)

Steps 1 and 2 can be interchanged. At any stage in the process it is possible to go from any step to any other step.

3.1.2 Acquisition prompted by enquiries

In this situation, an enquiry, typically by a user who is not a domain expert, either has no solution or a solution which is deemed to be inadequate (see Section xx below). A enquiry which has no solution is an enquiry for which there is no compatible region in the GKMS. This enquiry is then routed to a domain expert which uses it as a prompt for adding new knowledge (in the form of new mapping(s)) to the system. This process is illustrated in Figure 5, Part 2. The process in Figure 5 can be guided by the GKMS as follows:

1. GKMS presents a list of unanswered queries to a domain expert
    - The unanswered enquiries are presented as a view, with date, and reason for being unanswered .
    - The expert selects an enquiry and opens it.
2. The domain expert inspect the enquiry

- The enquiry is presented in the source context form (SCon1)
- The instruction on Scon1 is: "please inspect the enquiry and when ready press "next" to answer it". Scon1 cannot be edited.

3. The domain expert answers the query

- GKMS presents the destination context (Dcon1). Depending on the size of the display screen the two forms Scon1 and Dcon1 can be shown simultaneously. Dcon1 is editable.
- The instruction on Dcon1 is: "please enter the solution to the enquiry by specifying the relevant attributes and providing explanations for your answers.
- The expert defines the solution to the enquiry.
- The expert can proceed by pressing "next" or can go back to the enquiry (Scon1) by pressing "back". The only other option available to the expert is to abort the mapping definition process (by pressing the "cancel" button).
- Pressing "next" takes the expert to the generalisation process (see below)

4. Return to source context to generalise the enquiry to a region if possible

- GKMS presents Scon2. Scon2 is identical to Scon1 except that it can be edited.
- The instruction on Scon2 is: "Generalise the enquiry if possible by defining a region "around" the enquiry for which the answer specified on the previous screen applies". See Section 3.1.
- The expert can: a) go back to the Dcon1 to inspect or edit the solution just defined, b) press "next" to continue, or c) press "cancel" to abort the mapping process.
- Pressing "next" allows the knowledge item (or mapping) to be saved.

5. Save knowledge item

- GKMS presents the mapping form (Mform) which enables the expert to enter the title for the mapping and a summary. Only the title is mandatory. When finished the option is to press "next" or "back" or "cancel".
- With "back" the expert can go back to the previous display. "Cancel" aborts the knowledge mapping process.
- With "next" GKMS displays: "this knowledge item will be added to the knowledge base and will be used to answer future enquiries. OK, back or cancel".
- "OK" save the mapping, "back" goes back to the previous screen, "cancel" aborts the knowledge definition process.

When an enquiry has been answered in the way described above, it is taken off the list of unanswered queries and the view (see point 1 above) is updated.

At any time during steps described above, the expert can return to either the source context with the problem in it (no editing allowed), the editable source context with the region in it (to modify the region), or to the destination context to modify the solution proposed.

The process described above can be varied by exchanging steps 3 and 4, as shown below:

1. GKMS presents a list of unanswered queries to a domain expert
2. The domain expert inspects the enquiry
3. The domain expert generalises the enquiry to a region if possible
4. The domain expert answers the query
5. Save knowledge item

In a practical implementation the forms can all be presented to the expert, with only the one(s) that is (are) intended to be edited in the editing mode. The advantage of this is that the expert can always have a synoptic view of the knowledge item being defined.

3.1 ACQUISITION IN A VARIABLE CONTEXT

This Section recognises that mappings may have different source and destination contexts. The acquisition process for variable contexts builds on the acquisition process in a fixed context described in Section 3.1 above. The extra or modified steps are shown below in *italic*.

1. GKMS presents a list of unanswered queries to a domain expert

   - The unanswered enquiries are presented as a view, with date, and reason for being unanswered .
   - The expert selects an enquiry and opens it.

2. The domain expert inspect the enquiry

   - The enquiry is presented in the source context form (SCon1)
   - The instruction on Scon1 is: "please inspect the enquiry and when ready press "next" to answer it". Scon1 cannot be edited.
   - The instruction on Scon1 is: "please inspect the enquiry and when ready press "next" to answer it". Scon1 cannot be edited.
   - *The enquiry may be accompanied by a message from the user who filed the enquiry. The message may describe the enquiry more specifically or in more details (see Section xxx on "unanswered or poorly answered enquiries).*
   - *Based on this message or on professional judgement, the expert may decide to extend (or reduce) the source context. This is described in point 4b. below.*
   - *It is also possible to allow source context editing at this stage in the process:*
     - *Scon1 has an additional button "edit context"*
     - *pressing "edit context" allows Scon1 to be edited (the enquiry cannot be modified, the attributes in the enquiry cannot be deselected -the deselect option is added to Scon1, new attributes can be added.*
     - *see point 4b below for details.*
     - *Scon1, when shown at this stage in the process, has the message: "please*

modify context as appropriate without changing the enquiry. Press "next" when ready to answer it".

- *when finished editing the context the user can press "next" to move to the destination context to answer the enquiry.*

3. The domain expert answers the query

- GKMS presents the destination context (Dcon1). Depending on the size of the display screen the two forms Scon1 and Dcon1 can be shown simultaneously. Dcon1 is editable.
- The instruction on Dcon1 is: "please enter the solution to the enquiry by specifying the relevant attributes and providing explanations for your answers.
- The expert defines the solution to the enquiry.
- The expert can proceed by pressing "next" or can go back to the enquiry (Scon1) by pressing "back". The only other option available to the expert is to abort the mapping definition process (by pressing the "cancel" button).

3b. Specify appropriate destination context

This involves either reducing the context or enlarging the context. Reducing the destination context corresponds to considering fewer options for the outcome than there is present in Dcon1. Enlarging the context corresponds to adding new outcome options to Dcon1.

Dcon1 has one additional button labelled "add new attribute".

Reducing the context

- *By default all the attributes in Dcon1 are part of the mapping's context.*
- *The expert can deselect some of these attributes by "unchecking" it (each attribute can have a check box for selection purposes for example).*

Enlarging the context

- *Pressing the "add new attribute" button opens the context editing module (see Section 2 above).*
- *The destination context the expert can add to is the Dcon1.*
- Pressing "next" takes the expert to the generalisation process (see below)

4. Return to source context to generalise the enquiry to a region if possible

- GKMS presents Scon2. Scon2 is identical to Scon1 except that it can be edited.
- The instruction on Scon2 is: "Generalise the enquiry if possible by defining a region "around" the enquiry for which the answer specified on the previous screen applies". See Section 3.1.

4b. Specify appropriate source context

Reducing the source context Scon2 corresponds to considering fewer attributes for defining the enquiry and accepting fewer attribute which are not specified in the enquiry (that is, which do not play a role in the definition of the enquiry).Enlarging Scon2 corresponds to adding new attributes for enquiry definition and for specifying which of these attributes do not play a role in the enquiry.

Scon2 has one additional button labelled "add new attribute".

Reducing the context

- *By default all the attributes in Scon2 are part of the mapping's context.*
- *The expert can deselect some of these attributes by "unchecking" it (each attribute can have a check box for selection purposes for example).*

Enlarging the context

- *Pressing the "add new attribute" button opens the context editing module (see Section 2 above).*
- *The destination context the expert can add to is Scon2.*

4c. Review the region

- *The process is the same as in point 4. Above.*
- The expert can: a) go back to the Dcon1 to inspect or edit the solution just defined, b) press "next" to continue, or c) press "cancel" to abort the mapping process.
- Pressing "next" allows the knowledge item (or mapping) to be saved.

5. Save knowledge item

- GKMS presents the mapping form (Mform) which enables the expert to enter the title for the mapping and a summary. Only the title is mandatory. When finished the option is to press "next" or "back" or "cancel".
- With "back" the expert can go back to the previous display. "Cancel" aborts the knowledge mapping process.
- With "next" GKMS displays: "this knowledge item will be added to the knowledge base and will be used to answer future enquiries. OK, back or cancel".
- "OK" save the mapping, "back" goes back to the previous screen, "cancel" aborts the knowledge definition process.
- *Each knowledge item is saved with its source and destination context.*

At any time during steps described above, the expert can return to either the source context with the problem in it (no editing allowed), the editable source context with the region in it (to modify the region), or to the destination context to modify the solution proposed. In a practical implementation the forms can all be presented to the expert, with only the one(s) that is(are) intended to be edited in the editing mode.

4. KNOWLEDGE ACCESS

Interactive and non-interactive knowledge access is discussed in Section 3.7 and 3.8, Part 2. Here we consider the action which activates a knowledge search in an GKMS and the process used to presenting the outcomes of the knowledge items or mappings which are compatible with the enquiry.

4.1 TRIGGERING MECHANISMS

The access process can be triggered by a user who defines an enquiry or by a system enquiry, that is, an enquiry specified by values produced by another package or process in a computer system.

4.2 ACCESS MECHANISMS

In a similar way to the triggering mechanisms, access can be by presentation to a user on a screen (for explanation mappings) or by running some processes or modules (for action mappings). The way the final results are produced and presented to users is under the control of the processes or modules in the outcomes; it is not part of this description).

4.3 RANKING OF MAPPINGS AND THEIR OUTCOMES

Knowledge items or mappings need to be ordered according to their fit with the enquiries. This applies to definite outcomes only or to definites and candidates outcomes; that is to definite and candidate mappings. Rejected mappings are not considered here. The fit is determined by three factors:

1. Weight of each source attribute (weights may not all be equal).
2. For definite and candidate knowledge items: proportion of attributes in the enquiry which are satisfied by the region attributes (i.e. attributes belonging to the region) of the knowledge item.
3. For definite and candidate knowledge items: proportion of the attributes in the source context which belong to the region of a candidate knowledge item.

4.3.1 Weight of source attributes

a(i) attribute i
w(i) weight of attribute i
nw(i) normalised weight of attribute i
Sum w(i) over all attributes = Sw(i)
nw(i) = w(i) / Sw(i) normalisation is not strictly necessary

4.3.2 Proportion of enquiry attributes satisfied by region attributes

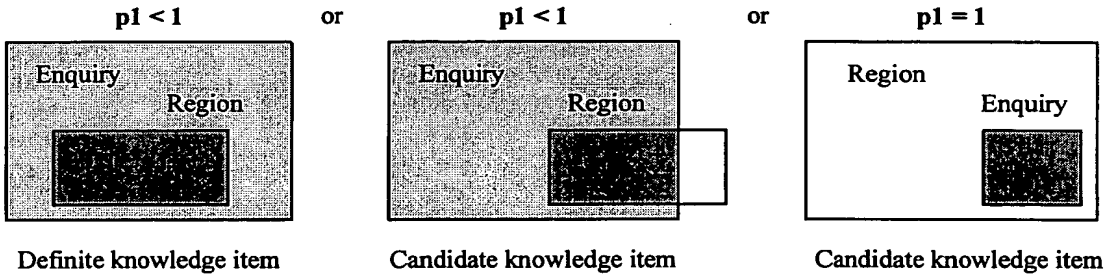
This specifies how general or specific a knowledge item is in addressing an enquiry. A high proportion 1/p_1 indicates a general knowledge item applicable to many enquiries. A low proportion indicates a well targetted answer. In practice is easier to use the inverse p1 of the proportion 1/p_1 (p1 <= 1). A value of p1 close to one indicates specific and precise knowledge and is highly desirable; a lower value is less desirable. This measure does not consider whether a knowledge item is a candidate or a definite (addressed in Section 4.3.3 below).

1/p_1 = Na(i,q) / (Na(i,r,q)) ; 1/p_1 >= 1
Na(i,q) = number of attributes i in enquiry q
Na(i,r,q) = number of attributes in the region r of a knowledge item which belong to the enquiry q

With attribute weights (see Section 4.3.1):

1/p1 = Nw(i,q) / (Nw(i,r,q)) ; 1/p1 >= 1

Figure 6: Illustrations for p1 <= 1 (attributes in region which "do not matter" not specified)



Definite knowledge item | Candidate knowledge item | Candidate knowledge item

A simple implementation of this ranking is the number of attributes in the region. Knowledge items which are found to be compatible with the enquiry and which have larger numbers of attributes are more specific than knowledge items with few attributes in their regions.

4.3.3 Proportion of a mapping's source attributes which satisfy the enquiry

This measure applies to candidate mappings. It determines how close candidate mappings are to becoming either rejected mappings or definite mappings. For definite mappings this proportion is 1. As for Section 4.3.2, the inverse of the desired proportion p2 is first calculated. p2 <= 1.

1/p_2 = Na(i,r) / (Na(i,r,q)) ; 1/p_2 >= 1
Na(i,s) = number of attributes i in the region r of a knowledge item
Na(i,r,q) = number of attributes in the region r of a knowledge item which belong to the enquiry q

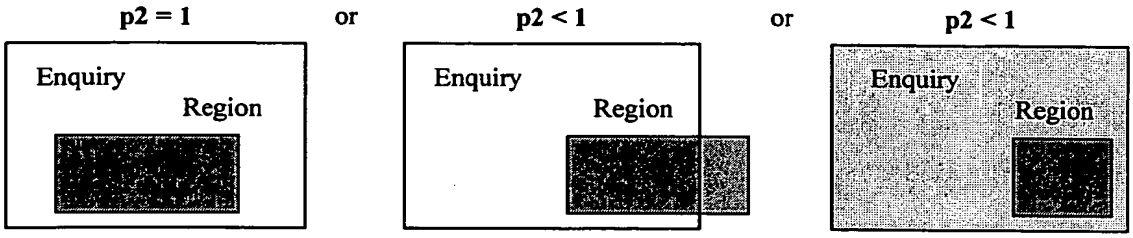
With attribute weights:

1/p2 = Nw(i,r) / (Nw(i,r,q)) ; 1/p2 >= 1

p2 = 1 for definite knowledge items

p2 < 1 for candidate knowledge items

Figure 7: Illustration for p2 <= 1 (attributes in region which "do not matter" not specified)



Definite knowledge item | Candidate knowledge item | Candidate knowledge item

4.3.4 Overall ranking

The relevance of a knowledge item, which determines its order or ranking among the knowledge items compatible with an enquiry (candidate of definite knowledge items), is calculated as:

For definite knowledge items: R = p1
For candidate knowledge items: R = p1 * p2

A relevance ranking can also include the probability or reliability level p3 that the outcome of a knowledge item solves the situations that can be described in its region (see Sections 3.5.1 and 3.5.2, Part 2). In this case we have:

For definite knowledge items: R = p1 * p3
For candidate knowledge items: R = p1 * p2 * p3

4.3.5 Related issues

Fuzzy logic principles can also be included in GKMS. For example, a query can cover several overlapping regions of several knowledge items. Similarly, outcomes can overlap. The impact on relevance can be calculated taking into account the overlaps and the weights of the attributes and similar considerations as in Sections 2 and 3 above.

4.4 UNANSWERED OR POORLY ANSWERED ENQUIRIES

An unanswered enquiry has no compatible mappings and therefore no outcomes. A poorly answered enquiry can be detected in several ways, which can be specified by users or domain experts:

- The definite knowledge items (and therefore outcomes) have a poor relevance ranking.
- There are candidates knowledge items but no definite knowledge items.
- The user deems the outcome as unsatisfactory and sends feedback to the GKMS administrator.

Once detected, GKMS deals with unanswered or poorly answered enquiries by referring them to an expert who can then add new knowledge in the GKMS to: a) answer this specific enquiry, and b) answer any future enquiry which falls within the region of the newly created knowledge item. See Section 3.1.2 for the knowledge acquisition process.

5. KNOWLEDGE MANAGEMENT FOR DOMAIN EXPERTS

Domain experts need tools to enable them to manage the knowledge items or mappings which have been entered into a GKMS. This becomes necessary as soon as more than a few mappings have been entered.

Domain experts need to manage the knowledge base from the point of view of knowledge comprehensiveness, consistency, quality, etc. The tools available to assist them are:

- Detect overlapping knowledge items, in their source and/or destination
- Identify knowledge items with a specified combination of source and destination attributes or values
- Edit existing knowledge items
- Inspect history of knowledge items in the system
  - list of knowledge items (active and deactivated)
  - when created, by whom

5.1 OVERLAPPING KNOWLEDGE ITEMS

Given a certain knowledge item, this tool detects which other knowledge items or mappings in the GKMS overlap with it. The overlap can be in the regions of the mappings or in their outcomes. This tool enables the domain expert to check that overlapping knowledge items are compatible.

With overlapping regions, an enquiry in the overlapping part or the regions will produce more than one outcome (one outcome per overlapping region). These outcomes need to be compatible (that is, not contradictory) for the system to provide meaningful results.

With overlapping outcomes, the domain expert can determine which conditions (i.e. regions) produce these outcomes and determine whether these different conditions are not mutually exclusive or contradictory for example.

If overlapping knowledge items are found to be incompatible in some of the ways described above, then the domain expert(s) need to edit one or several of them.

5.1.2 Overlap detection in fixed common contexts

Here we consider all knowledge items or mappings to have the same source and destination contexts. Note that we assume that the regions are defined by attributes which have specified values. The attributes which do not matter are not included in the region definitions (see Section 3.5.1, Part 2 for details).

Region overlap detection

The process for region overlap detection is as follows:

1. The expert selects which knowledge item to use as reference for detecting overlaps.
2. When the expert click the button "detect overlap", the GKMS:
   - Takes the region of the reference knowledge item.
   - Transforms this region into an enquiry (that is, GKMS displays it in the enquiry specification environment).
   - Activates the search for knowledge item (the same search as when searching for outcomes compatible with an enquiry; it searches for knowledge items with regions which are compatible with the enquiry, see Section 3.8.2, Part 2, for details). The reference knowledge item is not included in the list of knowledge items searched.
3. The GKMS search engine retrieves and presents the knowledge items with outcomes which are definite or candidates. This is different from a normal search in which the system only presents the outcomes).
4. The expert can then inspect and edit these knowledge items, see Section 3.5, Part 2.

Outcome overlap detection

The process for outcome overlap detection is as follows:

1. The expert selects which knowledge item to use as reference for detecting overlaps.
2. When the expert click the button "detect overlap", the GKMS:
   - Takes the outcome of the reference knowledge item.
   - Transforms this outcome (or region in the destination space) into an enquiry (that is, GKMS displays it in the enquiry specification environment).
   - Activates the search for knowledge item (the same search as when searching for outcomes compatible with an enquiry; it searches for knowledge items with outcomes (not regions in the source space as in a normal search) which are

compatible with the enquiry, see Section 3.8.2, Part 2, for details). The reference knowledge item is not included in the list of knowledge items searched.

3. The GKMS search engine retrieves and presents the knowledge items which are definite or candidates. This is different from a normal search in which the system only presents the outcomes).
4. The expert can then inspect and edit these knowledge items, see Section 3.5, Part 2.

In practice, experts often search for knowledge items which overlap in their sources and destinations

5.1.3 Overlap detection in variable contexts

Here we consider all knowledge items or mappings which can have different source and destination contexts. Not treated here.

5.2 SEARCH FOR KNOWLEDGE ITEMS WITH SPECIFIC CHARACTERISTICS

Experts can search the knowledge base of the GKMS for knowledge items which meet specified characteristics. The process is as follows:

1. The GKMS presents the expert with the source and destination contexts.
2. The expert defines an query in these contexts by specifying values for attributes in one or both of the contexts.
3. Pressing "detect overlap" activates the search for overlapping items. The search algorithm combines the two searches described in Section 5.1. That is the GKMS searches for both region and outcome overlaps.
3. The GKMS presents the knowledge items retrieved.
4. The expert can then inspect and edit these knowledge items, see Section 3.5, Part 2.

5.3 EDIT EXISTING KNOWLEDGE ITEMS

The knowledge editing process is described in Section 3.5, Part 2.

5.4 INSPECT HISTORY OF KNOWLEDGE ITEMS

The expert can inspect the list of knowledge items or mappings in the GKMS with the author status, date defined, and date deactivated (if appropriate) for each item.

Part 4: GKMS Applications and Examples

1. EFFLUENT DISPOSAL ADVISORY SYSTEM

In this Section we consider the desirable features and the design of a SSPG-based system for implementation as a computer system. To assist in this process we consider as example an application related to the discharge of effluents in a river system.

The methodology for achieving the design features mentioned is presented. The architecture is also discussed.

The design paradigm for the SSPG-based system is object oriented and hierarchical.

1.1 DOMAIN SPACE DEFINITION

This is controlled by the domain experts. They decide the type of applications they wish to design the system for. In our example, is it a system for the discharge of effluents in a river, in a natural lake, in the ocean or in all of them; does one need to consider all types of effluents or only those that are non-radioactive for example?

Once the type of applications is decided, the domain experts must specify the problem space and the solution space.

Problem space specification: Consists of the identification of all the key factors (or attributes) that enter into the definition of an enquiry related to the application field. In our example, some key factors could be: amount and concentration of a restricted substance in the effluent to be discharged and frequency of discharge.

Solution space specification: Consists of the identification of the components that could be used to define an outcome. In our example, it could include: discharge as proposed, dilute before discharging, process the effluent before discharging (so as to remove some highly toxic elements), etc. An outcome would then be a logical combination of these components.

It is difficult to define the problem space and the solution space once and for all. One can expect that some new factors will have to be added to the problem space and new components to the solution space. This is typically the case as experience is gained through the use of the system and the development or refinement (by domain experts) of application knowledge. In our example, for the problem space it could be the discovery of the existence of an endangered rodent near the proposed point of discharge or information about a new artificially produced compound that is dangerous to some fish. For the solution space it could be a new option afforded by a new policy developed by the

relevant authority or by some new technology (for the former it could be that discharge is only allowed when flow is above a certain value, and for the latter it could be to install a new type of filter before discharging into the river system).

It is also conceivable that the type of applications to be considered may need to evolve over time. Discharge into estuaries could be seen as both an extension of discharge into a river system or discharge into the ocean. Any such extension would impact on the domain

All changes to the domain space are recorded (see Section 5.8).

1.1.1 Problem space specification

The domain expert identifies the classes for the main attributes or factors that define the domain and the type of applications associated with it. We refer to these classes as level 0 classes. In our example, the level 0 classes are a) the effluent characteristics (including the characteristics of the restricted substances the effluent may contain), b) discharge characteristics, c) the characteristics of the water the effluent is proposed to be discharged into, and d) the characteristics of the environment around the proposed point of discharge.

Level 0 classes contain level 1 classes and objects. Each level i+1 class or object describes a feature of a level i class. Table 8 shows a possible definition of levels 0, 1 and 2 classes for our application:

Table 8: Levels 0, 1 and 2 classes for a hypothetical effluent discharge applications

Level 0 classes & objects	Level 1 classes & objects	Level 2 classes & objects
Effluent characteristics	Type	Oil or grease Organic substance Pesticide Weedicide Does not matter
	Form	Liquid Settleable solid Has floatable matter in it Has strings (>2m) in it Does not matter
	Restricted substances in it	Select items from the separate restricted substances list (see below) Does not matter
	pH	Number Does not matter
	Concentration	Number Does not matter
	Biochemical O2 demand	Number Does not matter

	Colour	Red Blue or black Green Does not matter
	Odour	Stagnant Sewer Does not matter
	Temperature	Number Does not matter
	Effluent characteristics do not matter	
Discharge characteristics	Amount	Number Does not matter
	Duration	Less than 1 hour 1 to 2 hours 2 to 5 hours 5 to 24 hours 1 to 2 days continuous Does not matter
	Frequency	Less than once per year 1 to 2 per year 1 to 4 per month 1 to 2 per day more than 2 per day Does not matter
	Discharge mode	Stormwater drain Shoreline or nearshore drain Deepwater (depth 2 to 5 m) Deepwater (depth > 5 m) Does not matter
	Discharge characteristics do not matter	
Water characteristics	Flow	Number Does not matter
	Temperature	Number Does not matter
	Water usage	Potable water supply Human bathing & swimming Irrigation & other public uses Animal consumption Does not matter
	Monitoring mode	Monthly measurements Weekly measurements Daily measurements

		Hourly measurements Does not matter
	Is a sensitive aquatic environment?	Yes No Does not matter
	Is water temperature monitored?	Yes No Does not matter
	Is water flow monitored?	Yes No Does not matter
	Is water pH monitored?	Yes No Does not matter
	Is water chemical composition monitored?	Yes No Does not matter
	Water characteristics do not matter	
Surrounding characteristics	Human habitations within 100 meters of proposed discharge point?	Yes No Does not matter
	Protected fauna or flora within 100 meters of the proposed discharge point?	Yes No Does not matter
	Surrounding characteristics do not matter	

Sometimes an object may refer to another class as shown for "restricted substances in the effluent" in the above Table (under level 0 object "effluent characteristics). This relates to the separate class of restricted substances which lists the relevant properties of the restricted substances.

Table 8a: Separate class dealing with restricted substances only

Level 0 classes & objects	Level 1 classes & objects	Level 2 classes & objects
Restricted substances characteristics	Names	Maximum permissible concentration in water
		Other characteristics

Level 1 objects are descriptive, they are attributes that describe the level 0 objects. Level 2 objects describe the values of the level 1 objects. The domain expert can modify level 1

and 2 objects. Level 1 objects result in question for a user and level 2 objects describe the choices the user has for its answers. When selected in a level (i) class, the "... does not matter" object has the effect of rendering inactive the corresponding object in the previous class (level (i-1) class).

Domain space is built using a database management system, with the domain specification operations corresponding to adding records (re: objects) linked to fields (classes).

Domain space specification can be carried out by domain experts as it does not require any specialist computer knowledge, but only domain knowledge.

1.1.2 Solution space specification

The solution space is defined as a level 0 class which comprises 2 other classes, as shown in Table 9.

Table 9: Solution space items for the hypothetical effluent discharge example

Level 0 classes & objects	Level 1 classes & objects	Level 2 classes & objects
Solution options	**External outcomes** (outcomes that do not affect objects in any other class in the system)	Discharge as proposed Do not discharge Discharge when river flow > x Carry out following changes Reduce effluent temperature to 10 degrees Neutralise chemical with y Install monitoring equipment Then discharge allowed etc
	Internal outcome (outcomes that can affect objects in other classes in the system)	

Domain experts can modify level 2 objects which they then use as a list from which to select the appropriate combination as outcome to an enquiry or a region. While it may not be strictly necessary for domain experts to define level 2 objects (as they have the expertise to specify answers without referring to a list), it is very convenient to have the list of approved options to select from. It is of significant importance when some options are related to regulations or government policies for example. In this case, some options may be determined more by legal considerations than by the domain of applications (in our example effluent and water chemistry) and may require a different type of experts. This illustrate that several experts in different domains can independently contribute to the knowledge in a system.

The internal outcomes mentioned in the Table can be troublesome as they affect objects that are being taken into account to define the outcome (there is a risk of creating some "circular effect" that are difficult to predict and often difficult to control). Avoiding circular effects is an important aspect of software engineering that seeks to contain effects to local modules or objects. It is interesting to note that, despite the preceding statement, circular effects are the "reasoning paradigm" of rule-based system (Section 2.3). This explains some of the problems associated with rule-based systems (they are notoriously difficult to test, therefore knowledge verification is a problem). This issue is treated again in Section 6.3 on "hierarchical and modular system architecture".

1.2 DOMAIN SPACE EDITING

Alterations to either the problem or the solution space need to be done easily by domain experts. When a change is being made, one needs to consider whether previously defined knowledge items (regions and associated outcomes) are still valid or whether they need to be edited. Several options exist:

1. Declare all previously defined knowledge items invalid.

   This is usually not recommended as much knowledge that is potentially still useful would be removed.

2. Declare all previously defined knowledge items unaffected by the change and still valid.

   In this case no knowledge is lost. This option is reasonable when a change to the domain space is prompted by a new enquiry that falls outside the current domain and requires its extension. The domain experts had certified the previous knowledge items and changes to the domain space do no affect the validity of the previous outcomes.

3. Review the knowledge base.

   This option applies when new knowledge has been acquired by domain experts about the domain of applications of the system, which requires changes to the domain space. In this case all previously defined regions could be affected. The knowledge base (the knowledge items) need to be checked, see Sections 5.5 and 5.6 below on knowledge base editing.

All changes to classes and objects related to the problem and solution spaces are logged.

1.3 ENQUIRY SPECIFICATION

A user specifies an enquiry by selecting the appropriate objects in the level 2 classes.

At enquiry specification time, the level 2 "... does not matter" objects are replaced by "...don't know". It is often advantageous to treat Level 1 objects that are not specified (left blank) as having "...don't know" values.

1.4 KNOWLEDGE DEFINITION

Domain experts define knowledge items (regions and outcomes), either prompted by enquiries or based on their experience (corresponds to hypothetical enquiries). This needs to be done by domain experts directly, without the need for a knowledge engineer as intermediary (see illustration in Figure 1, Section 2.1). This point is most important for reasons of convenience and cost. Domain experts and users must be able to use the system without logistics problems. Domain experts must feel they own the system and that they can use it effectively and treat it as an everyday tool.

1.4.1 Knowledge item specification

Region definition:

Region definition (the "first" component of a knowledge item) corresponds to specifying the objects in level 1 and 2 classes. Table 10 below gives an example, with the selection in ***italic bold***.

Table 10: Region definition for the hypothetical effluent discharge example, illustrated here for the effluent and discharge characteristics only (similar for the other level 0 classes)

Level 0 classes & objects	Level 1 classes & objects	Level 2 classes & objects
Effluent characteristics	Type	Oil or grease ***Organic substance*** ***Pesticide*** ***Weedicide*** Does not matter
	Form	Liquid Settleable solid Has floatable matter in it Has long strings (>2m) in it ***Does not matter***
	Restricted substances in it	***Arsenic, copper*** Does not matter
	pH	Number ***Does not matter***
	Concentration	***From 0.01 to 0.1 mg per litre*** Does not matter
	Biochemical O2 demand	Number ***Does not matter***
	Colour	Red Blue or black Green ***Does not matter***

	Odour	Stagnant Sewer ***Does not matter***
	Temperature	Number ***Does not matter***
	Effluent characteristics do not matter	
Discharge characteristics	Amount	Number Does not matter
	Duration	Less than 1 hour --- --- Does not matter
	Frequency	Less than once per year --- --- Does not matter
	Discharge mode	Stormwater drain --- --- Does not matter
	Discharge characteristics do not matter	

The Table reads as follows:

The outcome attached to this region is valid for all enquiries:

IF Effluent type is organic substance or pesticide or weedicide
AND effluent form is any one in the list (re: valid for all effluent forms in the list)
AND restricted substance is either arsenic or copper
AND for all possible effluent pH
AND for all possible effluent concentrations
AND effluent colour is any one in the list (re valid for all effluent colours in the list)
AND effluent odour is any one in the list (re valid for all effluent odours in the list)
AND for all possible effluent temperatures
AND discharge characteristics are any one in the lists (re valid for all discharge characteristics mentioned in the lists)
Etc

Domain experts define regions by specifying ranges for all level 0 & level 1 objects, by:

a. Selecting one or several items in a list

b. Specifying a range using numbers (ie: alpha <= range <= beta, with alpha and beta being ordered lists)

c. Selecting Yes, No or don't know in logical choices

d. Specifying that the previous level class does not matter

Region selection can be easily achieved by domain experts as it does not require any specialist computer knowledge, only domain knowledge.

Outcome definition:

The outcome (the "second" component of a knowledge item) is specified by selecting an appropriate logical combination from the level 2 outcome objects. If a desired option is missing, the domain expert (if accredited to do it) adds the desired option to the existing list. For example an outcome could be:

a) Carry out the following changes
b) Neutralise chemical with process y
c) Install concentration monitoring equipment
d) Then discharge allowed

1.4.2 Knowledge item specification and rule definition

The non-technical process presented above for knowledge definition corresponds to the definition of rules in traditional rule-based expert systems. In the example above, the rules implicitly defined are:

```
IF    ((effluent type is organic substance or pesticide or weedicide)
      OR    (effluent type is pesticide)
      OR    (effluent type is weedicide))
AND   ((restricted substance is arsenic)
      OR    (restricted substance is copper))
AND   (0.01 <= concentration <= 0.1)
THEN  (Carry out the following changes:
            (Neutralise chemical with process y
            And Install concentration monitoring equipment)
            Only then discharge allowed))
```

The cumbersome definition of rules as shown above is replaced by simple selections as shown in Table 3. In addition the process illustrated in Table 3 corresponds to specifying quickly a large number of rules or clauses in rules. The process presented moves knowledge definition from the field of the computer expert to the domain expert.

It is important to note that the rules above are not equivalent to the selection stated in Table 3 with associated outcome. The rules above say nothing about effluent form, pH, concentration, colour, odour and temperature, and about all the discharge characteristics. In contrast, Table 3 makes very precise statements about these factors. It would be possible to extend the rules above to cover all the information in Table 3 but it would be lengthy and most inconvenient. Indeed, the rules above are typical of rules in expert systems where knowledge is incomplete and the domain of applicability of the rule is not specified (rules are global in nature). Table 3 and its associated outcome "contains" the rules above and states the environment in which the rules can apply.

Table 3 and its associated outcome clearly specify some knowledge and the domain of applicability of this knowledge. It is done by domain experts, without recourse to a difficult computer language and notation and without assistance from a knowledge engineer.

Knowledge item definition can be either prompted by an enquiry or not prompted by an enquiry.

1.4.3 Knowledge specification prompted by an enquiry

In this case the domain expert specifies the answer to the enquiry and then defines a region around the enquiry, that is a region that contains the enquiry. The process is as explained above.

1.4.4 Knowledge specification not prompted by an enquiry

This takes place when domain experts wish to define rules which they deem to have general value for all types of enquiries that may be asked of the system in the future. Two possibilities exist:

a) The domain expert uses a hypothetical enquiry to define a knowledge item for this hypothetical enquiry. The process is as explained earlier.

b) The domain expert knows that some conditions conveniently expressed as conditions about some classes are of general importance in specifying valid regions and outcomes. For the effluent example, it may be that any effluent that has a restricted substance with a concentration larger than the maximum authorised value must not be discharge under any circumstances. This knowledge can be expressed in 2 ways:

   1. Using Table 1, the domain expert defines a region for each member of the class (re: for each restricted substance in the class) specifying the outcome for any concentration larger than the maximum authorised value (all other classes and objects are set as "does not matter"). This is region definition as explained above (Table 3).

   2. By defining a "cut off" region

Cut off region specification

A "cut off" region is a region that is used to set boundaries to the problem space. A cut off region is defined by using all the level 1 objects in one single region. For example, for restricted substances, the domain expert wishes to specify an outcome for all concentrations that exceed the maximum permissible concentration. The domain expert uses a modified version of Table 1a to define regions for the class, as illustrated below:

Table 11: Definition of "cut off regions" for a specific outcome

Level 0 class &	Level 1 class &	Level 2 class &	Level 3 class &

objects	objects	objects	objects
Restricted substances characteristics	Arsenic	Maximum permissible concentration in water	Concentration larger than max permissible
	Copper	Maximum permissible concentration in water	Concentration larger than max permissible
	Silver	Maximum permissible concentration in water	Concentration larger than max permissible
	etc	Maximum permissible concentration in water	Concentration larger than max permissible

The level 3 class specifies the values for one characteristic of the level 1 objects, here the level 2 object "maximum permissible concentration in water".

Table 11 reads:

IF Arsenic concentration is larger than its maximum permissible
OR IF Copper concentration is larger than its maximum permissible
OR IF Silver concentration is larger than its maximum permissible
etc

Then the outcome is the specific outcome associated with this region. This method is a convenient way of defining "cut off" regions, that is, regions that limit the size of the problem space for the definition of other regions. Note that a numerical value has to be entered by the domain expert for each member of class level 2 (for example, cut off may be at less than the maximum permissible concentration in some situations).

In some cases the list of objects in a class is so large that even the method above is time consuming. In this situation, one uses a variation of the method where the domain expert enters a symbol that states that a boundary of the region exists for all level 1 objects when one of their characteristics (a level 2 object) exceeds or reaches a certain value. In our example, it is when the concentration for each restricted substance exceeds the maximum permissible value. The region is defined as a list of IF and OR IF statements as in the previous method and the system transforms it into the appropriate expression.

Level 0 class & objects	Level 1 class & objects	Level 2 class & objects	Level 3 class & objects
Restricted substances characteristics	For all objects	Maximum permissible concentration in water	Concentration larger than max permissible

Two possibilities exist with cut off regions:

a) Cut off regions override other regions (default setting)
The answer to an enquiry that falls within a cut off region and another overlapping region is the outcome of the cut off region (the other outcome is discarded).

b) Cut off regions "co-operate" with other regions
The answer to an enquiry that falls within a cut off region and another overlapping region is treated in Section 5.4.5.

Region definition is a local process and situation b) above should be the exception. That means that domain experts do not need to consider whether other regions are relevant to the problem at hand. They can concentrate on using their domain knowledge to specify outcome and define regions

1.4.5 Overlapping regions

When a new region is being defined, two situations can arise:

1. The new region does not overlap with any of the previously defined regions.

   No further action is required.

2. The new region overlaps with some previously defined regions.

   This is likely to be the most frequent occurrence when the system has been used for a while and a significant amount of knowledge entered in it. The fact that regions overlap does not imply a problem per se. A domain expert may be confident and define a reasonable large region of equivalence around a specific enquiry. Later, a new enquiry falls just outside the region and prompts the domain expert to define another region which then overlaps with the previous one. The outcomes of these two regions may be the same or may differ. Several options exist:

   a) Accept the regions as valid but define a strategy as to which region to select (that is which outcome to use as answer) for enquiries that fall within the overlapping parts of overlapping regions (see Section 5.7 below). This is the most likely option selected by domain experts.

   b) Accept the regions as valid and take as outcome for the intersection of the overlapping region, a logical combination (usually the union) of the outcomes of the regions. (This option has to be treated with care as it may invite domain experts to define incomplete knowledge items, that is, knowledge items that provide only part of the answer to an enquiry.)

   c) Inform the domain expert that the new region overlaps with existing regions and presents these regions and outcomes. The domain expert then has the option of modifying the new region or some old regions. See Sections 5.4 and 5.5 below. The right to modify previously defined regions may depend on the degree of seniority of the domain experts involved. A relatively junior domain expert may not have the authorisation to modify a region defined previously by a more senior domain expert.

   d) A mix of a), b) and c) above where the domain expert has the option to modify some regions if desired or allowed, but where the processing strategy ensures that the system can operate with overlapping regions.

1.5 KNOWLEDGE INSPECTION AND EDITING

1.5.1 Knowledge inspection

Knowledge inspection refers to the ability to review the regions and the outcome separately from an enquiry. This facility is available to domain experts and users who are not domain experts. This allows the system to be used as a training aid for a trainee domain expert for example. Inspection does not give the right to modify knowledge items. Inspection can be based on:

- Outcomes. In this case the query is based on one or a logical combination of the components that make up the solution space (Section 5.1). The user is presented with the list of components in the solution space.
- Attributes. The query is based on the one or a logical combination of attributes (or factors) that make up the problem space (Section 5.1). The user is presented with the list of attributes in the problem space.
- Experts names. The user can inspect what knowledge items have been specified by whom. The user is presented with the list of domain experts who have contributed to the knowledge base.
- Dates. The user can inspect what knowledge has been defined when.
- Enquiries. The user may wish to inspect the enquiries that prompted the definition of knowledge items. Enquiries can be specified by date or client name.
- A logical combination of some of all of the above options.

In all cases, the systems presents all the enquiries, regions and outcomes that are compatible with the parameters of the enquiry.

1.5.2 Knowledge editing

Knowledge editing facilities enable domain experts to modify some knowledge items in the knowledge base. Knowledge editing operates jointly with knowledge inspection (Section 5.4 above). All the facilities listed in the previous Section are available to the domain expert who uses them to select which knowledge items may require editing. Overlapping knowledge items may be presented automatically to the domain expert by the system when some overlapping regions are detected, see Section 5.4).

Domain experts have access to a facility similar to the one used for knowledge definition (Section 5.4) and they can change the region (by changing one or several attributes in the problem space) or the outcome (by changing one or several components of the solution space or their logical combination).

Knowledge inspection and editing features as explained above can be implemented using standard database management techniques. In all cases, it corresponds to presenting the relevant fields and records, side by side when comparisons are required, to give domain experts the opportunity to correct regions and outcomes if they see fit or to remove knowledge items if required.

1.6 KNOWLEDGE PROCESSING

Processing here refers to the search for answers to a specific enquiry. This corresponds to finding the regions in the knowledge base that "contain" the enquiry. If none exist, the system does not have the knowledge to provide an answer to the enquiry. If there are more than one region the search strategy must decide which outcome to take into account. When this is done the outcome(s) corresponding to the selected region(s) is presented to the user as answer to the specific enquiry.

Domain experts need to be able to specify the processing strategy. This implies that the strategy must be expressed in non-computer terms so that domain experts can understand the options available and specify them as they see fit. The search options are search without inferencing and search with inferencing.

1.6.1 Region search without inferencing

This consists of searching for regions by i) defining the enquiry as completely as possible and ii) finding the regions that contain it. Region search can be carried out:

a) In the reverse chronological order with the first region found accepted.

b) In the chronological order with the first region found accepted.

c) By accepting all the regions and taking the union of their outcome as discussed in Section 5.4. (This option has to be treated with care as it may invite domain experts to define incomplete knowledge items, that is, knowledge items that provide only part of the answer to an enquiry.)

d) Based on a hierarchy of domain experts (regions by the most senior expert searched first, followed by the second most senior expert, etc), with the first region found accepted.

e) Based on dates. That is only regions defined after, before or between certain dates are considered as candidates for selecting the outcome to the enquiry.

f) Using a combination of the options presented above.

If no region is found, the user is notified and the enquiry is referred to a domain expert who will produce and answer for this enquiry and define a region of equivalence. Region and outcome make up a new knowledge item which is then available to future enquiries.

For a region to contain an enquiry, the region must contain the enquiry for all the objects (in level 2 class) that define the enquiry. The algorithm to determine whether a region contains the enquiry is as follows:

0. Initialise
   - Open list of candidate regions
   - Open enquiry
1. Go to Next candidate region
2. Go to next level 1 object in the enquiry
3. Take value for that object in the enquiry (take corresponding level 2 object)

4. Go to corresponding level 1 object in the region
5. Check the level 2 region object(s) associated with the level 1 object:
   - If it is a logical value
     check whether the value is identical to the value of the enquiry (the "don't know" enquiry value is "contained" by the "does not matter" value in the region)
   - If it is a numerical range
     check whether the range contains the value of the enquiry
   - If the level 2 object is a list
     check whether it has as a member an object with the value of the enquiry
6. If No, this region does not contain the enquiry
   - Go to 9
7. If Yes, this region contains the enquiry object, go to 8
8. If there is another object in enquiry, go to next object (go to 2)
   If there is no other object in the enquiry, the region is a candidate region, add region to the list of regions containing the enquiry
9. If there is another candidate region, go to next region (go to 1)
   If there is no other candidate region, Exit

There are many other ways to implement the algorithm which are significantly more computationally effective.

1.6.2 Region search with inferencing

As outlined in Section 4.3, regions search with inferencing consists of asking the user only these questions about the problem that enable the system to arrive at a solution as quickly as possible. The way the inferencing is carried out is beyond what domain experts need to know. However they need to specify whether they want this processing used instead of the regions search without inferencing.

Each question in region search with inferencing elicits some features about the enquiry and direct the search towards the most relevant part of the problem space, using the "discriminating power" of the attribute.

The discriminating power of an attribute is the ability of the attribute (more precisely the ability of the answer to a question aimed at determining the value of that attribute) to separate the regions in different categories. For example, in an effluent situation, the presence of protected fauna on the shore of the river near the proposed point of discharge may be important. To get the discriminating power, one counts the number of regions in each category (in our example, how many regions accept the presence of protected fauna on the shore and how many do not accept it).

An algorithm for region search with inferencing is:

0. Initialise
   - Open list of candidate regions, all regions are candidate

1. Calculate discriminating power ***d*** for each level 1 object (attribute), taking into account all (remaining) candidate regions.
   - for level 1 objects with logical outcome (Yes, No or don't know):
     d = Sum (regions with outcome Yes) / number of regions
   - for objects with ranges:
     d = Sum (regions inside range) / number of regions
   - for objects with lists:
     d = Sum (***di***) / number of items in list, where
     di is the discriminating power of item i in the list, defined as:
     di = Sum (regions with item i) / number of regions
2. Select the object with the highest discriminating power, that is, with ***d*** nearest to 0.5
3. Discriminate the regions, that is, present the question corresponding to the level 1 object with the highest discriminating power.
4. Use the answer to the question to cull the list of potential regions. That is, to remove all regions that do not contain the answer from the list of potential regions.
5. Check the list of remaining candidate regions
   - If there is no region left, there is no answer to this enquiry, go to 7.
   - If there is one region left, go to 6.
   - If there is more than one region left, go to 1.
6. Check that this region contains the enquiry
   6.1 Get list of level 1 objects in the region for which an answer has not yet been provided by the previous questions and that does not have as value (level 2 object) "does not matter":
   6.2 If no more such objects in the region, the enquiry is the answer. Go to 8.
   6.3 If such objects left in the region, go to 1.
7. Inform user that the system does not have the knowledge to answer the enquiry
   Answer to the enquiry has to be supplied by a domain expert
   Exit
8. Inform user that the enquiry has an answer and present the outcome of the region containing the enquiry as answer to the enquiry

Some modification to this algorithm are possible:

a) Check whether the candidate regions left have the same outcomes. If they have, there is no need to try to discriminate between these regions.
b) The algorithm terminates when a region has been found. It may be advantageous in some cases to find all the regions that contain the enquiry and then decide which one(s) to present (see Section 5.6.1).

There are other ways to implement the algorithm above which are significantly more computationally effective.

1.6.3 Comparison between search without and with inferencing

The choice of which technique to use depends on the complexity of the problem space (related to the number of attributes that are required to define an enquiry). If the number is large, one wishes to only ask the questions that are really necessary. Search with inferencing is then the preferred option. If on the other hand the number of attributes is not large and the effort required to answer the related questions is small then search without inferencing may be more convenient.

Search without inferencing enables users to play "what if" games effectively and conveniently. That is, the user can explore the relationship between values given to enquiry attributes and the outcome produced. It is achieved by the user defining the enquiry, getting the answer, reverting back to enquiry definition to change some attribute value(s), getting a new answer, etc. This may be very useful from a practical point of view.

1.7 EXPLANATIONS FACILITIES

The system enables domain expert to record explanations about the choices they make and the reasons for these choices. Explanations options are offered when experts:

- Define or edit the domain space (problem space and solution space)
- Define or edit knowledge items
- Set up the system (such as deciding on the processing strategies)

Explanations are then made available to users to explain the answers to the enquiries if requested. They also provide a very useful record of the progression of knowledge by these domain experts. Explanations about past decisions are important when domain experts review the knowledge in the system or when they need to edit this knowledge.

Explanation fields are attached to all commands available to domain experts and to all fields or record they define or edit.

1.8 AUDIT TRAILS

Audit trails can be kept for all uses of the system. They provide a complete history of the way the system has evolved over time and who has done what to and with it and when. It provides a time history of the development of the knowledge in the system and of the requirements placed on it during its life.

Audit trails are recorded for all actions that relate to:

- domain space definition or editing
- knowledge item definition or editing
- system setup (such as selection or change of search strategy)
- explanation facilities
- users

- enquiries

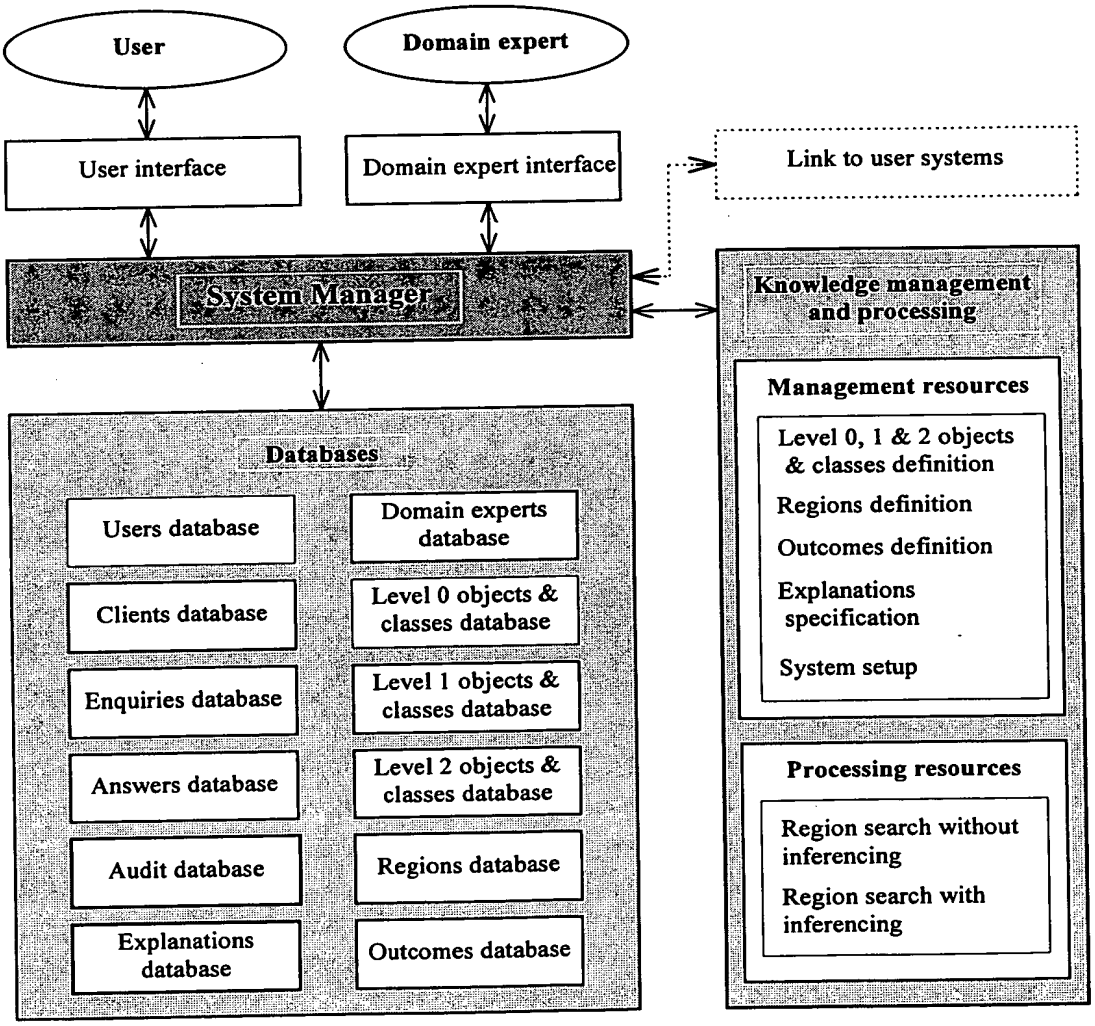
The system records names and dates and links them to the changes made and to the list of accredited domain experts.

Enquiries by users are also logged in audit databases.

1.9 ARCHITECTURE

The architecture required for supporting the features presented in the previous Sections is shown in Figure 8.

Figure 8: Architecture for a computer system based on the SSPG model of cognition



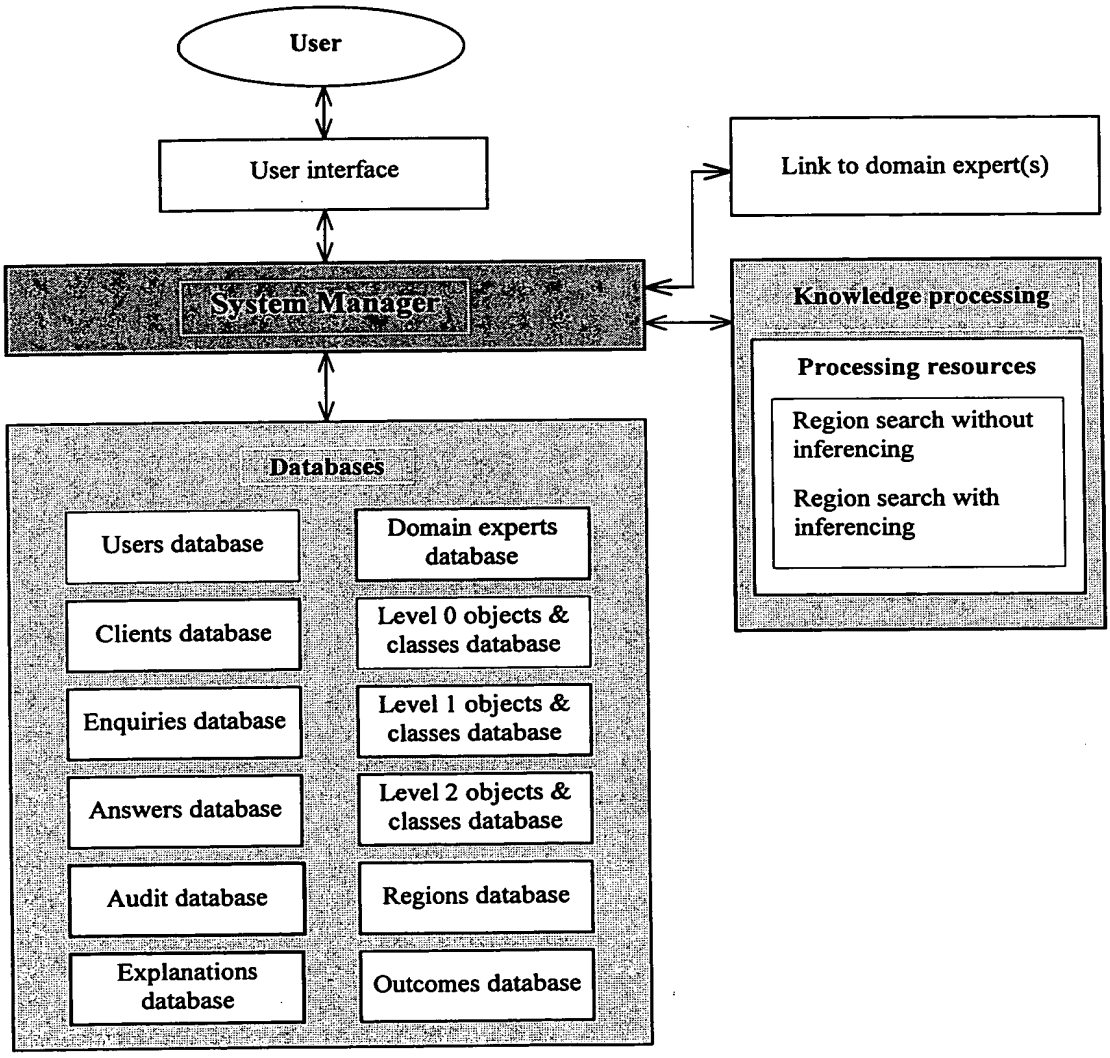
The architecture is for a system that can be used both by users and domain experts. In some situations, it can be very advantageous to have an architecture that separates

between user and domain expert functions. The user only system architecture is shown in Figure 9.

The user only system has access to all the databases, either for data input (ie: log client details, enquiry, etc) or for data access (ie: answer to enquiries and explanations).

The system may not have all the knowledge to answer all the enquiries, assistance from a domain expert may be required to extend the knowledge in the system. Therefore a link between user system and domain expert system is required. This link can be across a local or wide area network, and the databases for the user system may either be at a central site (controlled by a domain expert) or local in which case the domain expert need to be able to update them. The links are shown in Figures 8 and 9.

Figure 9: Architecture for a user only system based on the SSPG model of cognition



1.9.2 Implementation of effluent management system

The system described in this section has been developed for the purpose of testing the system and the development methodology.

Inspection of Figures 8 and 9 show that a significant part of the system relates to the database management. It is therefore natural to use a database management system as part of the development and implementation tools. The tools used were Microsoft Access and Microsoft Visual Basic. Other tools such as Lotus Notes or Internet technology can be used.

The architecture is as shown in Figure 8. The databases used and the knowledge management and processing functions are as shown in Figure 6 and as discussed in the previous Sections. The level 0,1 & 2 classes and objects are those presented in the previous Tables. The system manager was implemented as distributed logic supporting the Microsoft Access database management functions in one version and as a centralised module in Visual Basic in another.

2. LEARNING AND TRAINING SYSTEMS

This Section describes the implementation of GKMS for learning and training. GKMS can support the work of trainers and learners.

2.1 TRAINERS

Trainers can use the source context to:

- order and describe all the issues that need to be learned in a course or syllabus
- define scenarios or questions that learners needs to consider

Trainers can use the destination context to:

- order and present the material to be learned in a course or syllabus
- provide solution to scenarios or questions

GKMS give trainers a flexible environment for ordering their material and giving learners interactive access to this material. In effect, GKMS is a knowledge-base which contains the expertise of the trainers and the course material, and which can also provide links to supporting material.

2.2 LEARNERS

Learners can explore the knowledge base by:

- specifying enquiries in the source space. These enquiries bring all the knowledge items related to the enquiries, with explanations and supporting material. This use of GKMS is similar to the use by domain experts who need to perform knowledge management functions (see Section 5, part 2 for details)
- submitting their outputs in the destination space to questions or scenarios specified by trainers in the source context.
- checking their outputs to questions or scenarios against the outcomes recommended by the trainers

GKMS empowers learners by letting them explore the knowledge base at their own initiative. GKMS gives learners access to the material in a question-answer mode (or consultative mode).

Paul GUIGNARD and

SAVANT SYSTEMS INTERNATIONAL PTY LTD

REFERENCES

1. AI Expert. Expert System Resource Guide. AI Expert. December 1992
2. Booch G. Object Oriented Design. The Benjamin/Cummings Publishing Company Inc. 1991.
3. Boose J H. A survey of knowledge acquisition techniques and tools. Knowledge Acquisition. 1, 3-37. 1989.
4. Buchanan B G, Barstow D, Bechal R, Bennett J, Clancey W, Kulikowski C, Mitchell T, Waterman D A. Constructing an expert system. In Building Expert Systems, eds Hayes-Roth F, Waterman D A, Lenat D B. Addison-Wesley, 127-168. 1983.
5. Buchanan B G. Smith R G. Fundamentals of expert systems. Annu. Rev. Comput. Sci. Vol 3. 1988.
6. Buchanan B G, Bobrow D, Bavis R, McDermott J, Shortliffe E H. Knowledge-based systems. Annu. Rev. Comput. Sci. Vol 4. 1990.
7. Case-Based Reasoning and its Applications. Special issue of Expert Systems with Applications. Vol 6, No 1. 1993.
8. Chien C-C, Ho C-S. A primitives-based generic approach to knowledge acquisition. Knowledge Acquisition 6, 215-242. 1994.
9. Compton P, Kang B, Preston P, Mulholland M. Knowledge acquisition without analysis. European Knowledge Acquisition Workshop. Springer Verlag. 1993.
10. Devedzic V, Velasevic D. Features of second-generation expert systems - an extended overview. Eng. Appl of AI. Vol 3. 1990.
11. Eloranta K T. Case memory support systems: tools for computer supported case-based reasoning. In Advances in Information Modeling and Knowledge Bases. Eds: Jaakkola H, Kangassalo H, Ohsuga S. IOS Press. 1991.
12. Gruber T R. The Acquisition of Strategic Knowledge. Academic Press, 1989.
13. Harmon P, Maus R, Morrissey W. Expert Systems Tools and Applications. John Wiley & Sons, Inc. 1988.
14. Hickman F. Knowledge acquisition: the key to success for commercial expert systems. Proceedings of the International Conference on Knowledge-based Systems, 205-214, 1986.
15. Honavar V, Uhr L. Symbol processing systems, connectionist networks and generalised connectionist networks. Technical report no 90-23. Department of Computer Science, Iowa State University, Ames. 1990.
16. Jafar M, Bahill A T. Interactive verification of knowledge-based systems. IEEE Expert, Feb 1993.
17. Johnson-Laird P N. Human experts and expert systems. In Intelligent Systems in a Human Context: Development, Implications and Applications. Eds Murray L A & Richardson J T E. Oxford University Press, 35-46. 1989.
18. Ketler K. Case-based reasoning: an introduction. Expert Systems with Applications. Vol 6. 1993.
19. Kingston J. Progamatic KADS: a methodological approach to a small knowledge-based system project. Expert Systems. Vol 9, No 4. 1992.
20. Kilodner J. Retrieval and organizational strategies in conceptual memeory, a computer model. Ph.D. diss. Yale Univ. 1980.

21. Kolodner J, Cullingford R. Towards a memory architecture that supports reminding. In proceedings of the Eighth Annual Conference of the Cognitive Science Society. Amherst, Mass.: Cognitive Science Society. 1986.
22. Kolodner J. Case-Based Reasoning. Morgan Kaufmann Publishers, Inc. 1993.
23. Lee S, O'Keefe R M. Developing a strategy for expert system verification and validation. IEEE Transactions on Systems, Man and Cybernetics. Vol 24, No 4. 1994.
24. Levesque H J. Knowledge representation and reasoning. Ann. Rev. Comput. Sci. 1:255-287. 1986.
25. Mechitov A I, Moshkovich H M, Olson D L. Problems of decision rule elicitation in a classifaction task. Decision Support System. 12, 115-126. 1994.
26. Mettrey W. A comparative evaluation of expert system tools. IEEE Computer. February 1991.
27. Mott S. Case-based reasoning: market, applications and fit with other technologies. Expert Systems with Applications. Vol 6. 1993.
28. Mozer M C. RAMBOT: a connectionist system that learn by example. In IEEE First International Conference On Neural Networks, Vol II. Eds Caudill M & Butler C, 693-700. 1987.
29. Nwana H S, Bench-Capon T J M, Paton R C, Shave M J R. Domain-driven knowledge modelling for knowledge acquisition. Knowledge Acquisition 6, 243-270. 1994.
30. Olson D L, Mechitov A I, Moshkovich H M. The role of rules and examples in the process of knowledge acquisition in direct classification tasks. Expert Systems with Applications. Vol 8, 203-212. 1995
31. Preston P, Compton P, Kitkouhi D. An expert system interpreter for time course data with refinement in context.
32. Riesbeck C, Bain W. A methodology for implementing case-based reasoning systems. Lockheed. 1987.
33. Schank R. Conceptual dependency: a theory of natural language understanding. Cognitive Psychology. Vol 3, No 4. 552-631.
34. Schank R. Conceptual information processing. Fundamental Studies in Computer Science. Vol 3. North-Holland. 1975.
35. Schank R, Kolodner J. Retrieving information from an episodic memory, or why computers memories should be more like people;s. Technical report 159. Dpt of Computer Science. Yale Univ. 1979.
36. Seifert C, Hammond K J, Johnson H M, Converse T M, McDougal T F, Vanderstoep S W. Case-based learning: predictive features in indexing. Machine Learning. Vol 16. 1994.
37. Sestito S and Dillon T S. Automated Knowledge Acquisition. Prentice Hall, 1994.
38. Slade S. Case-based reasoning: a research paradigm. AI Magazine. Spring 1991.
39. Special issue: Verification and validation of intelligent systems: five years of AAAI workshops, part 1. International Journal of Intelligent Systems. Vol 9, No 4. 1994.
40. Tulving E. Episodic and semantic memory. In Organisation of Memory; eds Tulving E & Donaldson W. 381-403. 1972.
41. Tulving E. Elements of Episodic Memory. Oxford University Press.
42. Vargas J E, Raj S. Developing maintainable expert systems using case-based reasoning. Expert Systems. Vol 10, No 4. 1993.